INTERNATIONAL STANDARD

ISO 9506-1

First edition
2000-08-15

Industrial automation systems — Manufacturing Message Specification —

Part 1: Service definition

Systèmes d'automatisation industrielle — Spécification de messagerie industrielle —

Partie 1: Définition des services

Reference number
ISO 9506-1:2000(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Figures

Tables

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO 9506 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 9506-1 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 5, *Architecture, communications and integration frameworks*.

This first edition of ISO 9506-1 cancels and replaces ISO/IEC 9506-1:1990, of which it constitutes a technical revision. It incorporates the corrections published in ISO/IEC 9506-1/Cor.1:1995 and in ISO/IEC 9506-1/Cor.2:1995, the additional services published in ISO/IEC 9506-1/Amd.1:1993, and in ISO/IEC 9506-1/Amd.2:1995, and the material published in ISO/TR 13345.

ISO 9506 consists of the following parts, under the general title *Industrial automation systems — Manufacturing Message Specification*:

— *Part 1: Service definition*
— *Part 2: Protocol specification*

Annexes A to C form a normative part of this part of ISO 9506. Annexes D to F are for information only.

Introduction

This part of ISO 9506 provides a wide variety of services useful for various manufacturing and process control devices. It is designed to be used both by itself and in conjunction with Companion Standards that describe the application of subsets of these services to particular device types.

The services provided by the Manufacturing Message Specification (MMS) range from simple to highly complex. It is not expected that all of these services will be supported by all devices. The subset to be supported is limited in some cases by Companion Standards, and in all cases may be limited by the implementor. Characteristics important in selection of a subset of services to be supported include:

a) applicability of the service to the device;

b) the complexity of services and requirements;

c) the complexity of provision of a particular class of service via the network versus the complexity of the device.

Security considerations

When implementing MMS in secure or safety critical applications, features of the OSI security architecture may need to be implemented. This International Standard provides simple facilities for authentication (passwords) and access control. Systems requiring a higher degree of security will have to consider features beyond the scope of this International Standard. This International Standard does not facilities for non-repudiation.

Complexity of services and requirements

Some MMS services are quite complex and should be considered advanced functions. Devices used in very simple applications often will not require such advanced functions, and hence will not support such MMS services.

Keywords

Application Interworking
Application Layer Protocol
Information Processing Systems
Manufacturing Communications Network
Manufacturing Message Specification
Numerical Control System
Open Systems Interconnection
OSI Reference Model
Process Control System
Programmable Controller
Programmable Device
Robotics Control System
Virtual Manufacturing Device

General

This part of ISO 9506 is one of a set of International Standards developed to facilitate the interconnection of information processing systems. It is positioned within the application layer of the Open Systems Interconnection Environment as an Application Service Element (ASE) with respect to other related standards by the Basic Reference Model for Open Systems Interconnection (ISO 7498).

The aim of Open Systems Interconnection is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of information processing systems:

a) from different manufacturers;

b) under different managements;

c) of different levels of complexity;

d) of different ages.

Purpose

The purpose of this part of ISO 9506 is to define the services provided by the Manufacturing Message Specification. The Services specified in this part of ISO 9506 are realized by the Manufacturing Message Specification Protocol specified in ISO 9506-2, making use of services available from the underlying communication system. Although using the model in ISO 7498, this International Standard is written to be independent of the exact form of the communication system insofar as possible. This International Standard does this by describing its support requirements as a set of abstract services present in a variety of communication environments. Realization of these abstract services in an OSI environment is described in Annex A.

This part of ISO 9506 is concerned, in particular, with the communication and interworking of programmable manufacturing devices. By using this standard together with other standards positioned within the OSI Reference Model, otherwise incompatible systems may work together in any combination.

ISO 9506-2 specifies the protocol that supports the Manufacturing Message Specification.

Edition

This part of ISO 9506 differs from ISO/IEC 9506-1:1990 in the following ways.

a) The informal object modelling used in ISO/IEC 9506-1:1990 has been replaced by the use of the object modelling techniques present in ASN.1, ISO/IEC 8824-2. Hence, this part of ISO 9506 defines an ASN.1 module, MMS-Objects-Module-1, that contains the object models on which the service procedures are based.

b) The material in ISO/IEC TR 13345 that specifies subsets of protocol for MMS has been used in this part of ISO 9506 to specify options within the object models.

c) All the material of Amendments 1 and 2 have been incorporated into the document, as well as the Technical Corrigenda.

d) The services and protocol present in the Companion Standards already published, ISO/IEC 9506-3, ISO/IEC 9506-4, ISO/IEC 9506-5 and ISO/IEC 9506-6, have been incorporated into the base standard, and new parameter CBBs have been added to the Initiate procedure to indicate their presence. The concept of Companion Standard has been simplified to a document that makes explicit the relationship between the abstract models in MMS and the requirements of the application field that is the subject of the Companion Standard.

   As a result of this incorporation, the need for separate abstract syntaxes for each of the Companion Standards has been removed. All Companion Standards can now operate in the single abstract syntax of the base standard, although using other abstract syntaxes remains a possibility for backward compatibility.

e) The communication requirements of MMS have been generalized so that MMS is described with respect to an abstract set of services needed for its support. The relation between this abstract set of services and the services provided by the suite of OSI communication protocols is specified in an annex. This opens the possibility of having MMS operating correctly over alternate communication systems (such as reduced stack implementations) as long as the equivalent of these abstract services is provided.

f) The restrictions on the characters that can be used as an Identifier have been relaxed to allow an Identifier to begin with a numeric character, and by extension, to consist solely of numeric characters.

g) Many (but not all) occurrences of VisibleString have been replaced by a new production MMSString that provides the option of using an extended latin alphabet, suitable for western Europe, and an option to use an arbitrary string of characters taken from ISO 10646 or from elsewhere.

h) A new service, ReconfigureProgramInvocation, has been introduced into the clause on Program Invocation management. This service provides a technique of dynamically changing the constituent Domains of a running Program Invocation.

i) A new field was added to the object model of the Named Variable and Named Type. This field may be used to describe the semantics associated with the Named Variable or Named Type. The field is either predefined or has its value

established as the name of the Named Type used to construct it in the DefineNamedVariable or DefineNamedType service. This field can be reported with the GetVariableAccessAttributes or GetNamedTypeAttributes service if a new parameter CBB has been negotiated.

j) The material of the document has been reorganized to provide shorter clauses.

k) The Real Data type has been removed from the document.

l) The Scattered Access has been removed from the base document and placed in an informative annex.

m) In accordance with the recommendations in ISO/IEC 8824-1, all occurrences of EXTERNAL in the protocol have been replaced with CHOICE { EXTERNAL, EMBEDDED PDV }.

Protocol

Because of the use of the ASN.1 object modelling technique, the protocol now exists in two separate modules, one that is part of the object model contained in this part of ISO 9506, and a second module defined in ISO 9506-2 that describes the content and structure of all valid PDUs. Despite the fact that the ASN.1 formulation appears different in some cases, nevertheless the PDUs produced through application of ISO/IEC 9506 are identical with those produced by this edition. For this reason, this edition continues to be identified by the major version number one. (The minor version number has been changed to reflect all the new additions to the document.)

There are two exceptions to this statement that should be noted.

a) Syntactic extensions defined by the companion standards are now identified by new parameter CBBs instead of a separate abstract syntax. Therefore, for any use of MMS involving companion standard facilities, there is a change in the Initiate PDU. However, if the companion standard facilities are not used, the Initiate PDU remains the same as that defined by the first edition.

b) Some small changes have been made to the tagging in the ChangeAccessControl service (part of Amendment 2) to bring it into alignment with corresponding protocol in the GetNameList and Rename services.

ASN.1 Modules

The ASN.1 modules defined in ISO 9506 may be obtained from the SC 4 Secretariat in computer readable format. The modules are available in two forms: as published and with the IF - ENDIF brackets removed.

To obtain these files use the Internet location: <http://forums.nema.org:8080/~iso_tc184_sc5>.

Industrial automation systems - Manufacturing Message Specification - Part 1: Service definition

1 Scope

The Manufacturing Message Specification is an application layer standard designed to support messaging communications to and from programmable devices in a Computer Integrated Manufacturing (CIM) environment. This environment is referred to in ISO 9506 as the manufacturing environment. This part of ISO 9506 does not specify a complete set of services for remote programming of devices, although provision of such a set of services may be the subject of future standardization efforts.

This part of ISO 9506 defines the Manufacturing Message Specification within the OSI application layer in terms of

a) an abstract model defining the interaction between users of the service;

b) the externally visible functionality of implementations conforming to ISO 9506, in the form of procedural requirements associated with the execution of service requests;

c) the primitive actions and events of the service;

d) the parameter data associated with each primitive action and event;

e) the relationship between, and the valid sequences of, these actions and events.

The service defined in this part of ISO 9506 is that which is provided by the Manufacturing Message Specification protocol. The service may be used by other application layer service elements or by other elements of the application process.

This part of ISO 9506 does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. This part of ISO 9506 specifies the externally visible functionality of implementations together with conformance requirements for such functionality.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 9506. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 9506 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest eidition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange.*

ISO 7498:1984, *Information processing systems - Open Systems Interconnection - Basic Reference Model.*

ISO 7498-2:1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture.*

ISO 7498-3:1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 3: Naming and addressing.*

ISO/TR 8509:1987, *Information processing systems - Open Systems Interconnection - Service conventions.*

ISO 8571 (all parts), *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management.*

ISO/IEC 8649:1996, *Information technology - Open Systems Interconnection - Service definition for the Association Control Service Element.*

ISO/IEC 8650:1996, *Information technology - Open Systems Interconnection - Connection-oriented protocol for the Association Control Service Element: Protocol specification.*

ISO 8822:1988, *Information processing systems - Open Systems Interconnection - Connection oriented presentation service definition.*

ISO/IEC 8824-1:1995, *Information technology - Abstract Syntax Notation One (ASN.1). - Specification of basic notation.*

ISO/IEC 8824:1995-1/Amd. 1:1996, *Information technology - Abstract Syntax Notation One (ASN.1). - Specification of basic notation - Amendment 1: Rules of extensibility.*

ISO/IEC 8824-2:1995, *Information technology - Abstract Syntax Notation One (ASN.1) - Information object specification*

ISO/IEC 8824-2:1995/Amd. 1:1996, *Information technology - Abstract Syntax Notation One (ASN.1) - Information object specification - Amendment 1: Rules of extensibility*.

ISO/IEC 8825-1:1995, *Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

ISO 9506-2:2000, *Industrial automation systems - Manufacturing Message Specification - Part 2: Protocol specification.*

ISO/IEC 9545:1989, *Information technology - Open Systems Interconnection - Application Layer structure.*

ANSI/IEEE 754:1985, *IEEE Standard for Binary Floating-Point Arithmetic.*

3 Definitions

NOTE The definitions contained in this clause make use of abbreviations defined in clause 4.

For the purposes of this part of ISO 9506, the following definitions apply.

3.1 Reference Model definitions

This part of ISO 9506 is based on the concepts developed in the Basic Reference Model for Open Systems Interconnection (ISO 7498), and makes use of the following terms defined in that International Standard:

a) application-entity;

b) application-process;

c) application service element;

d) open system;

e) (N)-protocol;

f) (N)-protocol-data-unit;

g) (N)-service-access-point;

h) (N)-layer;

i) system;

j) (N)-user-data.

3.2 Service Convention definitions

This part of ISO 9506 makes use of the following terms defined in the OSI Service Conventions (ISO/TR 8509) as they apply to the Manufacturing Message Specification:

a) confirm;

b) indication;

c) primitive;

d) request;

e) response;

f) service primitive;

g) service provider;

h) service user.

3.3 Abstract Syntax Notation definitions

This part of ISO 9506 makes use of the following terms defined in the Abstract Syntax Notation One (ASN.1) Specification (ISO/IEC 8824-1):

a) value;

b) type;

c) simple type;

d) structure type;

e) component type;

f) tag;

g) tagging;

h) type (or value) reference name;

i) character string type;

j) boolean type;

k) true;

l) false;

m) integer type;

n) bitstring type;

o) octetstring type;

p) null type;

q) sequence type;

r) sequence-of type;

s) tagged type;

t) choice type;

u) selection type;

v) object identifier type;

w) module;

x) production;

y) ASN.1 encoding rules;

z) ASN.1 character set;

aa) external type.

3.4 Other definitions

This part of ISO 9506 makes use of the following terms:

3.4.1 AA-specific (Application Association specific):
an adjective used to describe an object whose name has a scope that is a single application association (i.e. the name may be referenced only on the application association with respect to which the object was defined).

3.4.2 attribute:
a data element, having a defined meaning, together with a statement of the set of possible values it may take.

3.4.3 Called MMS-user:
the MMS-user that issues the Initiate.response service primitive.

3.4.4 Calling MMS-user:
the MMS-user that issues the Initiate.request service primitive.

3.4.5 Client:
the peer communicating entity that makes use of the VMD for some particular purpose via a service request instance.

3.4.6 conformance building block (CBB):
an atomic unit used to describe MMS conformance requirements.

3.4.7 data:
any representation to which meaning is or might be assigned (e.g. characters).

3.4.8 domain:
an abstract object that represents a subset of the capabilities of a VMD that is used for a specific purpose.

3.4.9 Domain-specific:
an adjective used to describe an object whose name has a scope that is a single Domain (i.e. the name can be referenced over all application associations established with the VMD that may reference this Domain).

3.4.10 download:
the process of transferring the content of a Domain, including any subordinate objects, via load data to a remote user.

3.4.11 event management:
the management of event conditions, event actions, event enrollments, and event condition lists.

3.4.12 file:
an unambiguously named collection of information having a common set of attributes.

3.4.13 file operation:
the transfer of files between open systems, the inspection, modification or replacement of part of a file's content, or the management of a file and its attributes.

3.4.14 filestore:
an organized collection of files, including their attributes and names, residing at a particular open system.

3.4.15 information:
the combination of data and the meaning that it conveys.

3.4.16 journal:
a set of recorded, time-tagged event transitions, variable data, and/or comments, that may be logically ordered during retrieval.

3.4.17 local matter:
a decision made by a system concerning its behaviour in the Manufacturing Message Specification that is not subject to the requirements of ISO 9506.

3.4.18 Manufacturing Message Protocol Machine (MMPM):
an abstract machine that carries out the procedures specified in this part of ISO 9506.

3.4.19 MMS-environment:
a specification of the service elements of MMS and semantics of communication to be used during the lifetime of an application association.

3.4.20 MMS-provider:
that part of the application entity that conceptually provides the MMS service through the exchange of MMS PDUs.

3.4.21 MMS-user:
that part of the application process that conceptually invokes the Manufacturing Message Specification.

3.4.22 monitored event:
a detected change in the state of an event condition.

3.4.23 network-triggered event:
an event that occurs due to an explicit stimulus by a client.

3.4.24 operator station:
an abstract object representing equipment associated with a VMD that provides for input/output interaction with an operator.

3.4.25 predefined object:
an object that is instantiated through the use of some mechanism other than an MMS service.

3.4.26 Program Invocation:
an abstract object representing a dynamic element that most closely corresponds to an execution thread in a multi-tasking environment, composed of a set of Domains.

3.4.27 Receiving MMPM:
the MMPM that receives an MMS PDU.

3.4.28 Receiving MMS-user:
the MMS-user that receives an indication or confirmation service primitive.

3.4.29 Requesting MMS-user:
the MMS-user that issues the request service primitive for a service.

3.4.30 Responding MMS-user:
the MMS-user that issues the response service primitive for a service.

3.4.31 semaphore:
a conceptual lock associated with a logical or physical resource that permits access to that resource only by an owner of the lock.

3.4.32 semaphore management:
the control of semaphores.

3.4.33 Sending MMPM:
the MMPM that sends an MMS PDU.

3.4.34 Sending MMS-user:
the MMS-user that issuess a request or response service primitive.

3.4.35 Server:the peer communicating entity that behaves as an agent for a VMD for a particular service request instance.

3.4.36 standardized object:
an object instantiation whose definition is provided in this part of ISO 9506 or in an MMS Companion Standard.

3.4.37 upload:
the process of transferring the content of a Domain, including any subordinate objects, via load data from a remote user, in such a manner as to allow subsequent download.

3.4.38 variable:
one or more data elements that are referred to together by a single name or description.

3.4.39 variable access:
the inspection or modification of variables or components of variables defined at a VMD.

3.4.40 Virtual Manufacturing Device (VMD):
an abstract representation of a specific set of resources and functionality at a real manufacturing device and a mapping of this abstract representation to the physical and functional aspects of the real manufacturing device.

3.4.41 VMD-specific:
an adjective used to describe an object whose name has a scope that is a single VMD (i.e. the name may be referenced by all application associations established with the VMD).

4 Abbreviations

AA	application association
ACSE	Association Control Service Element
AE	application entity
AP	application process
ASE	application service element
ASN.1	Abstract Syntax Notation One
CBB	conformance building block
CIS	Configuration and Initialization Statement
FRSM	file read state machine
FTAM	File Transfer, Access and Management
MMPM	Manufacturing Message Protocol Machine
MMS	Manufacturing Message Specification
NC	Numerical Control
OSI	Open Systems Interconnection
PC	Programmable Controller
PDU	protocol data unit
PSAP	presentation service access point
SAP	service access point
SDU	service data unit
ULSM	upload state machine
VMD	Virtual Manufacturing Device
VT	Virtual Terminal

5 Conventions

5.1 Base of Numeric Values

This part of ISO 9506 uses a decimal representation for all numeric values unless otherwise noted.

5.2 Object modelling

This part of ISO 9506 makes use of a technique of abstract object modelling in order to describe fully the MMS device model and the MMS service procedures. In this modelling technique, abstract objects, the characteristics of such objects, and operations on those objects are described. The objects defined are abstract and aid in the understanding of the intent of MMS service procedures and their effects. In implementing MMS, a real system maps the concepts described in the model to the real device. Hence, as viewed externally, a device that conforms to this part of ISO 9506 exhibits the characteristics described in the object modelling technique, but the mechanisms for realization of this view are not defined by this part of ISO 9506.

MMS defines several classes of objects. Each object is an instance of a class; a class exhibits certain characteristics and may be affected by certain MMS services and operations. Each class is given a name, by which it may be referenced.

Each class is characterized by a number of attribute types that serve to describe some externally visible feature(s) of all objects of this class. Each instance of a class (object) has the same set of attribute types, but has its own set of attribute values. The values of these attributes are defined by this part of ISO 9506 or may be established by MMS services; hence a change in the device may be modeled by a change in one or more attribute values of an object (or objects).

Each object must be uniquely identified among all instances of the same class. For this purpose, one or more of the object's attribute values, as a combination, must be unique. (For example, many objects have an attribute type called "object name", which is different for each object of the same class.) In MMS, each attribute that is a part of this combination of attributes that make the object unique is identified as a "key attribute".

Some attributes of the objects only need occur if certain parameters are negotiated during association establishment. These parameters, exchanged during the association establishment procedure, are called conformance building blocks (CBB). While the basic object model remains unaffected, the exact details of the object model in use on any instance of communication depends on which of these parameters have been negotiated during association establishment.

Finally, some objects contain attributes that are conditional, in the sense that they are relevant to the object if and only if certain conditions (other than the conformance building blocks) hold true. MMS expresses such attributes through the use of a "constraint". Attributes that are subject to a constraint are considered to be object attributes for an object if and only if the corresponding constraint is satisfied for that object.

This International Standard makes use of the object modelling facilities of ASN.1, ISO/IEC 8824-2. The type language specified in ISO/IEC 8824-1 is used for describing the abstract structure of a protocol, that is, the data present in a message. It does so by providing a notation for data values and data types. A data value is an instance of a data type, and a data type can be thought of as a name for the set of possible data values for that type.

ISO/IEC 8824-2 extends this paradigm to introduce the concept of object and class. An object is an instance of class, and a class is defined by an abstract structure of a set of attributes, called fields. Each field is composed of two elements, a field identifier, and a field type. For the purposes of MMS models, the field can be restricted to one of four possible choices:

a) the field is a data value (an instance of a data type);

b) the field is a data value set (a subtype of a data type);

c) the field is an object (an instance of a class);

d) the field is an object set (a subset of a class).

The field identifier always begins with an ampersand (&) followed by a name. The use of capital and lower case letters serve to specify which choice is being used. The following example illustrates the four possibilities:

```
NewObject ::= CLASS {
    &invokeID              INTEGER,
    &Capabilities          MMSString,
    &selected-PI           PROGRAM-INVOCATION,
    &AbstractSyntaxes      ABSTRACT-SYNTAX
    }
```

a) &invokeID. The field identifier begins with a lower case letter (the 'i' following the ampersand) and the field choice is the name of a data type, INTEGER. This field is a simple data value field.

b) &Capabilities. The field identifier begins with an upper case letter and the field choice is the name of a data type. This field is a value set, one or more values from a specific type.

c) &selected-PI. The field choice identifies an object class, a PROGRAM-INVOCATION. By convention, all object classes are named using capital letters. The field identifier itself begins with a lower case letter indicating that the field is a single object from that class.

d) &AbstractSyntaxes. The field identifier begins with an upper case letter and the field choice identifies an object class. This field is an object set composed of one or more instances of the object class ABSTRACT-SYNTAX. (ABSTRACT-SYNTAX is an object class defined in an annex to ISO/IEC 8824-2.)

NOTE ISO/IEC 8824-2 allows other specifications for fields, but these are the only ones that need to be considered for MMS models.

5.2.1 References to other objects

Some objects contain attributes that identify other objects. In the example above, &selected-PI and &AbstractSyntaxes identify other objects. Such attributes, called reference attributes, provide a mechanism to create a linkage from one object to another. The method of representing such attributes in a real system is a local matter, and such attributes may not be directly modified or examined. Many MMS services provide the capability to determine the identity of an object referenced by such a linkage, however, through the use of such an indirect reference.

NOTE Reference attributes are similar to "pointer" types available in many programming languages. As an example of the operation of reference attributes, the MMS Rename service may be used to change the Object Name attribute of a referenced object, while having no effect on the value of the reference attribute.

In certain cases, reference attributes may be missing from the object model. A special object, the NULL object, is used to model this situation. Services that report this situation return the distinguished value UNDEFINED. The consequences of this value and its use on the behaviour of other objects is specified in the relevant object and service descriptions. Once a reference attribute becomes NULL, creation of an object of the same type as the object formerly referenced with the same attribute values does not change the reference attribute (it retains the value NULL, and reporting services continue to report UNDEFINED).

5.2.2 Field Types

Types for fields defined in this part ISO 9506 make use of types defined in ISO/IEC 8824-1, clause 3. Additionally, complex types are used that are constructed from the ISO/IEC 8824-1 primitive types and subsequently named to allow them to be referenced.

5.3 Specialisation of MMS

5.3.1 Conformance Building Blocks (CBB)

MMS prescribes a procedure, used at the time of association establishment, in which sets of parameters are exchanged for the purpose of identifying the services that may be performed during the association. The effect of identifying these parameters is such that the protocol available to be used during the association is limited to a proper subset of the entire protocol specified in ISO 9506-2. This same procedure affects the attribute structure of the object model operative for that instance of communication The parameters exchanged are of two types:

a) those that are announced by the two MMS users, the service conformance building blocks,

b) those that are negotiated between the two MMS-users, the parameter conformance building blocks. These CBBs are proposed by the Calling MMS-user and either accepted or rejected by the Called MMS-user. Negotiation always works to reduce the set proposed, never to augment it.

Declaration of support of any service CBB requires inclusion of the protocol related to that service in the protocol set to be used on that association. Support of a parameter CBB usually results in the inclusion of some optional fields within the protocol of some service request or response. However, in some cases, support of a parameter CBB implies support of one or more service CBBs, regardless of whether or not support for those service CBBs has been declared. The full effect of these CBBs on the protocol specification is given in ISO 9506-2. Here we note the effect of these CBBs on the object model.

5.3.2 Notation

The notation introduced has the form of a preprocessor language in which ASN.1 is embedded. It is very similar in concept to the macro preprocessor for the C language. There are only two commands used in this notation:

- IF (<list of arguments>)

- ENDIF.

The IF command requires an argument list enclosed in parentheses; the arguments are the names of the conformance building blocks, either service or parameter. One or more such arguments must appear. If there is more than one argument, the arguments are separated by one or more spaces. The argument is treated as a boolean variable that has the value true if the corresponding service or parameter building block is supported as a result of the MMS Initiate exchange. If there is one argument, the lines following the IF statement up to the matching ENDIF statement are to be included in the resulting ASN.1 definition if and only if the conformance building block so named in supported. If there is more than one argument, the lines following the IF statement are to be included if any of the conformance building blocks in the argument list is supported. (This can be thought of as a 'logical OR' function of the conformance building blocks.)

IF statements may be nested to any depth; the effect of

IF (x)

IF (y)

is to include the lines following these commands if and only if both x and y are true; that is, if conformance block x and conformance block y are both included. (This can be thought of as a 'logical AND' function of the conformance building blocks.)

The ENDIF statement is used to end the scope of an IF statement. Each IF statement must have a matching ENDIF statement.

NOTE This notation is a proper subset of the notation employed in ISO 9506-2 to define the proper protocol for the instance of communication.

5.3.3 Constraints

Constraints (other than occurrences of CBBs) are expressed in this object model by use of the ASN.1 construct OPTIONAL together with a comment that indicates the conditions under which this attribute is or is not included in the object model.

5.4 Service Parameter Description

This part of ISO 9506 uses a tabular format to describe the component parameters of the MMS service primitives. This same format is used to describe several complex parameters that are used in more than one service. Each table consists of up to six columns, containing the name of the service parameter, a column each for the request ("Req"), indication ("Ind"), response ("Rsp"), and confirm ("Cnf") primitives, and a column for conformance building block specification ("CBB"). The "Rsp" and "Cnf" columns are absent when the service is not a confirmed service. For parameter specifications, the Req and Rsp may be combined into a single column as may the Ind and Cnf.

5.4.1 Service Table Structure

For those tables that require support of particular parameter conformance building blocks, the required conformance building blocks are enumerated on the first line in the table. In the remainder of each table, one parameter (or part of it) is listed on each horizontal line. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the vertical column:

M - parameter is mandatory for the primitive
U - parameter is a user option, and may or may not be provided depending on dynamic usage by the MMS-user
C - parameter is conditional upon other parameters or the environment of the MMS-user
S - parameter is a selection from a collection of two or more possible parameters. The parameters that make up this collection are indicated in the table as follows:

a) each parameter in the collection is specified with the code "S";

b) the name of each parameter in the collection is at the same indentation from the beginning of the parameter column in the table;

c) Either
- **1) each parameter is at the leftmost (outer) indentation in the table; or**
- **2) each parameter is part of the same parameter group. A parameter group is a collection of parameters where each group member has a common parent parameter. The parent parameter for any group member is the first parameter above the member that is not indented as far as that member. In the example below, ParameterA and ParameterB form a parameter group:**

```
ParameterX
    ParameterA
    ParameterB
ParameterY
    ParameterC
```

Informally, for parameters involved in a selection, the indentation in the services tables signifies which parameters are involved in a selection. All parameters at the same level of indentation that are under a common "higher level" parameter are a part of the same selection.

The code "(=)" following one of the codes M, U, C or S indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. (For instance, an "M(=)" code in the indication service primitive column and an "M" in the request service primitive column means that the parameter in the indication primitive is semantically equivalent to that in the request primitive.)

Some parameters may contain subparameters. Subparameters are indicated by labelling of the parameter as M, U or C, and indenting all subparameters under the parameter. Presence of subparameters is always dependent on presence of the parameter that they appear under (for example, an optional parameter may have subparameters; if the parameter is not supplied, no subparameters may be supplied).

The CBB column is used to indicate that usage of the parameter is dependent on support of conformance building blocks, other than that containing the service. If no entry exists in the CBB column, there is no dependency on other conformance building blocks. If an entry does exist, the parameter is available (and permitted for use by this International Standard) if and only if the named conformance building block is supported and negotiated for use.

Some service parameters are named using a "List Of ..." convention. Unless otherwise noted, all parameters whose names begin with "List Of ..." specify a list of zero or more of the item specified after the "List Of" keyword phrase. (This type of parameter corresponds with the sequence-of ASN.1 type in ISO 9506-2.)

The descriptions of parameters in this part of ISO 9506 make reference to types, in order to describe the allowable values for such parameters. The types referenced may either be types defined in ISO/IEC 8824-1 or may be defined in ISO 9506-2.

5.4.2 Collating Sequences

Several object models contain fields that are themselves sets of other objects (more properly sets of references to other objects). When such sets are to be reported as parameters of a service response, the order of the elements shall be based on the &name field of the objects. The syntax of this &name field is defined by the Identifier type in clause 7.5.2 of ISO 9506-2. The Identifier type lists the permissible characters of the &name field, and gives an ordering of these characters. This ordering shall be used to define the collating sequence of the set of objects so named.

If objects of different name scopes are to be combined on a composite list, the objects of AA-specific scope shall appear first grouped by the application association, the objects of VMD specific scope shall appear next, and the objects of Domain scope shall appear last grouped by Domain. The ordering of the Domains is as specified in this clause; the ordering of the application associations is a local matter.

Some service parameters refer to arbitrary sequences of character strings (MMSString); in the cases in which a collating sequence is needed for these parameters, the character strings should be ordered using the prescriptions of Clause 36 of ISO/IEC 8824-1.

NOTE This clause does not provide an ordering for the CHARACTER STRING type.

5.5 Invocation Identifier on Service Primitives

For services identified in the ConfirmedServiceRequest production in clause 7.1 of ISO 9506-2, each MMS service primitive contains an "Invoke ID" parameter, which is mandatory in the request, indication, response, and confirm primitives. The value in the indication, response, and confirm primitives is semantically equivalent to that in the request primitive. This parameter serves to identify unambiguously the service invocation from an MMS-user on an application association. This parameter is not explicitly shown in the service primitive tables, nor is it explained separately for each service.

5.6 List Of Modifier on Service Primitives

Every confirmed MMS service contains a "List Of Modifier" parameter, which is a user option in the request and indication primitives. The value in the indication primitive is semantically equivalent to that in the request primitive. This parameter serves to specify a list of one or more service state machine modifiers which add a condition that must be satisfied for the execution of the service request to begin. This parameter is not explicitly shown in the service primitive tables, nor is it explained separately for each service.

MMS defines two modifiers: the AttachToSemaphore modifier and the AttachToEventCondition modifier, which are described in clauses 16 and 18, respectively.

The effect of the modifier on the Transaction state machine for execution of a confirmed MMS service is described in clause 7.

5.7 Addressing in MMS

This International Standard does not provide the means for naming and addressing of a peer MMS-user or peer Manufacturing Message Protocol Machine (MMPM). It makes use of the addressing and naming facilities in the underlying communication system. Specifics about the use of naming and addressing within OSI may be found in Annex A.

5.8 Service Conventions

This part of ISO 9506 uses the descriptive conventions contained in the OSI Service Conventions (ISO/TR 8509). The OSI Service Conventions define the interactions between the MMS-user and the MMS-provider. Information is passed between the MMS-user and the MMS-provider by service primitives, which may convey parameters. The following apply to the use of this model:

a) ISO/TR 8509 defines a model for the service provided by a layer of the OSI Reference Model. The MMS service does not correspond to such a layer (it describes a part of the application layer) but the model used is identical in all other respects;

b) at any instant in time, an application entity has multiple service requests outstanding, each proceeding independently of the others.

NOTE It should be noted that the MMS-user/MMS-provider distinction is an abstraction, and may not necessarily correspond to the realization of MMS in any particular system. Clauses 6 and 7 provide further details on the usage of abstract models.

5.9 Calling and Called MMS-user

This part of ISO 9506 makes use of the terms Calling and Called MMS-users. The Calling MMS-user is the MMS-user that issues the Initiate.request service primitive. The Called MMS-user is the MMS-user that issues the Initiate.response service primitive.

NOTE The use of the term "called" in MMS is not the same as the general usage of the term in OSI. The MMS usage of the term "called" corresponds to the OSI usage of the term "responding". This distinction has been introduced in order to avoid confusion with the Requesting/Responding MMS-user definition given below.

5.10 Sending and Receiving MMS-user and MMPM

This part of ISO 9506 makes use of the terms Sending and Receiving MMS-users. The Sending MMS-user is the MMS-user that issues a request or response service primitive. The Receiving MMS-user is the MMS-user that receives an indication or confirmation service primitive.

NOTE It is important to note that, in the course of completion of a confirmed MMS service, both MMS-users will be senders and receivers at one time. The first MMS-user sends the request and receives the confirmation, while the second MMS-user receives the indication and sends the response.

This part of ISO 9506 makes use of the terms Sending and Receiving MMPMs. The Sending MMPM is the MMPM that sends an MMS PDU. The Receiving MMPM is the MMPM that receives an MMS PDU.

5.11 Requesting and Responding MMS-user

This part of ISO 9506 makes use of the terms Requesting and Responding MMS-users. The Requesting MMS-user is the MMS-user that issues the request service primitive for a service, while the Responding MMS-user is the MMS-user that issues the response service primitive for a service.

NOTE It is important to note that the use of the term Responding MMS-user differs from the use of the term Responding entity in ACSE and other Standards. In those Standards, the term is used to reference the entity that responds to a connection request.

5.12 Client and Server of a Service

This part of ISO 9506 makes use of the terms Client and Server in order to describe the model of the MMS VMD (The VMD is described in clause 7). The Server is defined as the peer communicating entity that behaves as a VMD for a particular service request instance. The Client is the peer communicating entity that makes use of the VMD for some particular purpose via a service request instance. The VMD model is primarily useful in describing the actions of the Server, and thus in describing the commands and responses that a Client may use. A real end system may adopt the Client role, or the Server role, or both during the lifetime of an application association. Use of MMS in the OSI environment is further described in clause A.

Figure 1 depicts the relationships of the client and server of a service, the requesting and responding MMS-user, and the sending and receiving MMS-user and MMPM.

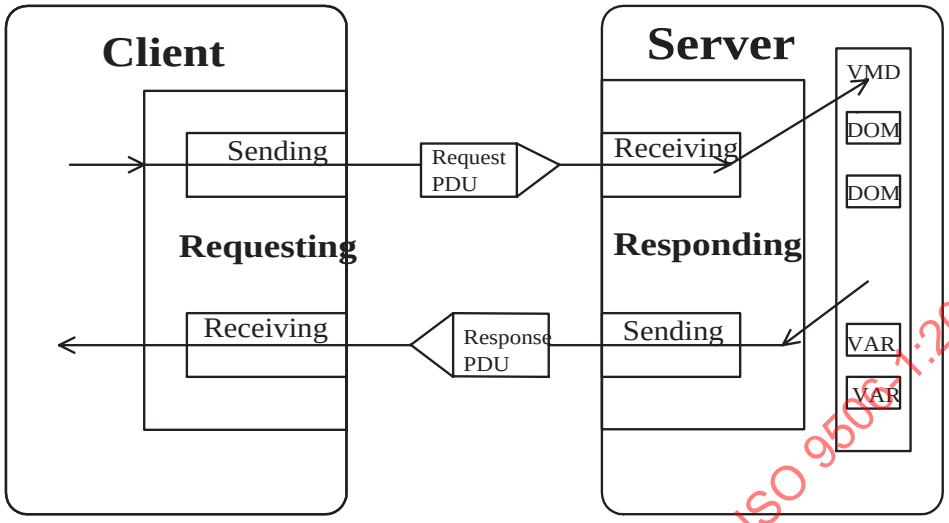

Figure 1 - Relationships of Client and Server, Requesting and Responding MMS-user, and Sending and Receiving MMPM

5.13 Relationship of Object Models to Service Tables

Throughout the text, the term "parameter" will be used to refer to components of a service primitive, and the term "attribute" or "field value" will refer to a component of an object model. In many cases, there is a one to one correspondence between a parameter in a service primitive and an attribute in an object model. However, in other cases the parameter is derived from one or more object attributes. When referring to a distinguished value of an attribute in an object model, a **distinctive type font** will be used. When referring to a distinguished value of a parameter of a service primitive, the value will be in upper case, e.g. UNDEFINED.

6 MMS in the OSI Environment

This clause describes the relationship between MMS and the communication environment that supports it. Although this International Standard is designed to be used in a variety of communication environments, it uses the model and terms of the OSI model (ISO 7498 and ISO/TR 8509) to describe its environment and the elements of the services needed to support it. Specific use of OSI communications is defined in annex A. Clause 7 describes the specific model of the MMS device within this environment.

Additional information to supplement this clause may be found in the OSI Reference Model (ISO 7498-1 and ISO 7498-3), the OSI Service Conventions (ISO/TR 8509), and the OSI Application Layer Structure (ISO/IEC 9545). References may be found in clause 2. This information may be particularly useful in understanding the relationship between AEs, APs, and their respective invocations.

The development of standards for the interconnection of manufacturing monitoring and control devices is assisted by the use of abstract models. To specify the externally perceived behaviour of interconnected manufacturing devices, each manufacturing device is represented by a functionally equivalent abstract model of the device called a Virtual Manufacturing Device (VMD) (see clause 7). The VMD model, along with the extensions to this model which are provided by clauses 7 to 23, describes the externally visible aspects of these devices.

In order to accomplish this, however, it is necessary to describe both the internal and external behaviour of these manufacturing devices. Only the external behaviour of the devices is retained as the standard of behaviour of a VMD. The description of the internal behaviour of such devices is provided in the model only to support the definition of the externally perceived aspects. Any manufacturing device that behaves externally as a VMD can be considered to be in conformance with ISO 9506.

NOTE The reader not familiar with the technique of abstract modelling is cautioned that the concepts introduced in the description of the VMD constitute an abstraction despite a similar appearance to concepts commonly found in real devices. Therefore, the VMD model is not a specification for an implementation.

6.1 Information Processing Tasks and Real Systems

The control and monitoring of a manufacturing process is a distributed information processing task. In order to carry out this task successfully, inter-operation of a number of real open systems is required. In OSI, a real open system is defined as a set of computers and associated software (including peripherals, terminals, human operators, physical processes, etc.) that follow the OSI Reference model.

In OSI, the inter-operation of real open systems is modelled in terms of the interactions between Application Processes (APs) in these systems. The distributed task of control and monitoring of a manufacturing process requires the co-operation of two or more Application Processes.

6.2 Application Processes

An Application Process (AP) is an element within a real open system that takes part in the execution of one or more information processing tasks. It is an abstract representation of those aspects of a real open system that are specific to the performance of those tasks. APs generally have requirements above and beyond any requirements to communicate with other APs. In the manufacturing environment, an AP represents a real system participating in the overall control/monitoring distributed task.

6.3 Interaction of Application Processes

Several requirements must be met in order to allow co-operative operation of APs. First, they must share sufficient information to enable them to interact and carry out processing functions in a compatible and co-operative manner. In the manufacturing environment, the term "universe of discourse" is used to name a model of those aspects of the "world" that are pertinent to the objectives of a manufacturing control and monitoring task of an AP. In order to allow successful interworking between APs, they must share a universe of discourse (i.e. they must have a common understanding of the manufacturing environment).

It is convenient to describe the common model of the manufacturing environment, or universe of discourse, in terms of a set of abstract "objects". Examples of objects in the manufacturing environment include variables, programs, and semaphores. The model describes the characteristics of objects and relationships between them. The characteristics may include the properties of those objects (either static or dynamic), and these properties are expressed as a set of rules and constraints about the behaviour of these objects within the model of the manufacturing environment. An example of a property of an object is the characteristic of a program that it is either running or stopped.

A universe of discourse can be described formally by a "conceptual schema". The conceptual schema for the manufacturing universe of discourse is described by the models provided by MMS. The second requirement for successful interworking of two APs is that they share a common model of the objects (such as variables, programs, and semaphores) in the universe of discourse. In OSI, this is called a "shared conceptual schema". This requirement is met in a manufacturing control and monitoring application through the sharing of the models in MMS for the manufacturing universe of discourse.

When two APs co-operate, their behaviour is determined partly by these models (the shared conceptual schema) and partly by their past interactions. The past interactions are modelled by the state of the objects in the universe of discourse. As an example, the co-operative behaviour of two APs may be partly determined by the Program Invocation model of MMS and partly by the state of Program Invocation objects in the APs. The shared information about the state of objects is called an information base.

6.4 Interaction of Application Processes in OSI

The activity of a given AP in a specific information processing task is supported by an application-process-invocation. At any time, an Application Process may have zero, one or more application-process-invocations. While the Application Process describes a specific set of information processing functions in a particular real system, the application-process-invocation (AP-invocation) describes an instance of the Application Process in a real system for a particular occasion of information processing. (Hence, an AP may describe control of a robot arm in a specific real system, while an AP-invocation describes the control of a robot arm in a specific real system on some occasion of control for assembly of some particular part).

Thus, co-operation between APs for the performance of a given information processing task takes place through interacting AP-invocations. When APs interact using communication capabilities, these communication capabilities are in turn modelled by one or more Application Entities (AE). An AE is an active element in the Application Layer of an open system, and it represents those parts of an Application Process involved in communications. Each AE represents exactly one AP.

Like APs, the activity of a given AE is supported by an application-entity-invocation. An AE-invocation performs the functions of an AE for a particular occasion of communication. At any time, an AP-invocation may be represented by zero or more AE-invocations for each AE associated with the AP. Hence, the interactions of AP-invocations for a particular occasion of communication is represented in OSI by a set of corresponding AE-invocations.

Because an AE describes only part of the operation of an AP (namely, that involving communications), resources in AP-invocations may exist beyond the lifetime of (and may be independent of) an AE-invocation.

NOTE The relationship between an AE and the VMD is defined in clause 7.

6.5 Structure of Application Entities

The AE represents a set of communication capabilities of an AP. These capabilities are defined by a set of Application Service Elements (ASEs). An ASE is a coherent set of integrated functions that provides a capability for communications for some particular purpose. MMS is modelled as an ASE for the purpose of communications with manufacturing devices.

6.6 Addressing of Application Entities

The underlying communication system uses some system of addressing and naming to identify nodes on the communication network and AE-invocations within each node. In the process of establishing an application association to support the MMS environment, the MMS-user must have knowledge of the addressing conventions used by the underlying communication. However, the specification of this addressing and naming conventions is outside the scope of this International Standard.

While in the MMS environment, an MMS user may need to refer to some communication node and/or AE-invocation other than the one with which the application association has been established. For this purpose this International Standard makes use of an "Application Reference" to identify an AE in another system. This is in the form of an ASN.1 type definition "ApplicationReference", appropriate to the underlying communication system. For use of MMS in an OSI based communication system, ApplicationReference is defined in annex A.

6.7 Application Context

The application context identifies the set of application service elements, their options, rules for use of the service elements, and their effects that are available on an application association. In the case of MMS, the application context associated with MMS indicates that the rules and requirements associated with ISO 9506 are in effect when the MMS application context is negotiated.

6.8 Presentation Context, Abstract Syntaxes, and Transfer Syntaxes

In OSI, there is an important distinction between the generic requirements of an application for the transfer of data and how those requirements are met in terms of a specific representation of data values. The former aspect of data description is referred to by the term "abstract syntax", while the latter aspect is referred to by the term "transfer syntax".

Abstract syntaxes are intimately associated with application protocol standards. ISO 9506-2 defines an abstract syntax matching its data transfer requirements. An abstract syntax can be viewed informally as describing the generic structure of data. In MMS, the set of type definitions provided in ISO 9506-2 constitutes an abstract syntax. The MMS abstract syntax may be supported by many different transfer syntaxes.

Transfer syntaxes are concerned with the way in which data is actually represented in terms of bit patterns during transmission. A transfer syntax may have attributes that are not related to the abstract syntaxes that it can support, but such attributes may be significant. For example, a transfer syntax may provide data compression or encryption. Such attributes (and how well they meet the requirements of the device) may influence the choice of syntaxes offered or selected for an instance of communication.

For the purpose of transferring data between MMS entities, it is necessary to identify the abstract syntax being used (MMS) and a transfer syntax that is capable of representing data values that are generated using this abstract syntax. A specific combination of an abstract syntax and a transfer syntax that is used for transfer of data is called a presentation context.

Abstract Syntax Notation One (ASN.1) is an example of a tool for specification of syntaxes and associated encoding rules. (The application of encoding rules to a particular abstract syntax may generate a transfer syntax.) The ASN.1 Specification (ISO/IEC 8824) has been chosen to describe the MMS abstract syntax. One possible transfer syntax for MMS is defined by the application of the ASN.1 Basic Encoding Rules (ISO/IEC 8825-1) to the MMS abstract syntax, but others are possible. The only requirement on a transfer syntax to be used for MMS is that it faithfully encode all of the elements of abstract syntax defined ISO 9506-2.

6.9 MMS requirements of the communication system

This International Standard requires that the underlying communication system provide the encoding mechanism (transfer syntax) to support the transfer of data expressed in the abstract syntax defined in ISO 9506-2. This International Standard requires the communication system to support full duplex, connection oriented peer to peer communication. In addition, this International Standard requires the support of the following abstract services from the communication system. The relationship of these services to OSI communication facilities is described in annex A, and the use of these services by MMS services is described in clause 24 of ISO 9506-2.

6.9.1 M-ASSOCIATE

The M-ASSOCIATE service allows an MMS-user to establish an association with another MMS-user over the communication system and to convey to the peer MMS-user a set of MMS specific parameters to be associated with this instance of communication. This service requires a set of parameters most of which are specific to the communication systems being employed. However, there are two parameters required of this service in support of other MMS functions, the identity of the other MMS-user, and (optionally) a password or other authentication parameter to be transmitted to the peer MMS-user. M-ASSOCIATE is a confirmed service. In addition to its other functionality, M-ASSOCIATE allows one MMS-user to transmit some user data (the service parameters of the MMS Initiate service) to its peer MMS-user. Upon completion of the service either (1) an association has been established and the MMS environment has been created (positive result parameter), or (2) the association could not be established (negative result parameter). Figure 2 shows the time sequence of service primitives for this service.

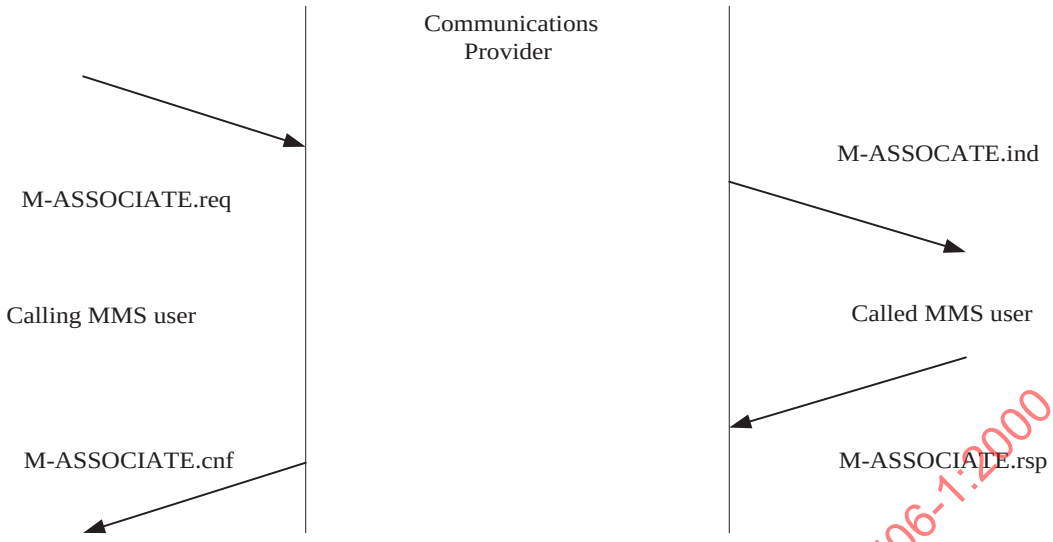

Figure 2 - M-ASSOCIATE service

Table 1 shows the parameters of the M-ASSOCIATE service.

Table 1 - M-ASSOCIATE service

Parameter	Req	Ind	Rsp	Cnf
Argument	M	M		
Calling Application Reference	M	M(=)		
Called Application Reference	M	M(=)		
Other Communication Parameters	M	M		
Authentication Value	U	U(=)		
User Data	M	M(=)		
Result			M	M(=)
Responding Application Reference			M	M(=)
Other Communication Parameters			M	M
Authentication Value			U	U(=)
User Data			M	M(=)

6.9.1.1 Argument

This parameter contains the parameters of the M-ASSOCIATE service of the Calling system.

6.9.1.1.1 Calling Application Reference

This parameter identifies the calling system uniquely within the communication system.

6.9.1.1.2 Called Application Reference

This parameter identifies the system with which the Calling system wishes to establish an association.

6.9.1.1.3 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.1.1.4 Authentication Value

This optional parameter may be included if the called system requires authentication to allow some services to be performed. It is used to establish the right of the calling system to make use of those services.

6.9.1.1.5 User Data

This parameter contains the MMSPdu as formed by the Initiate service request (see clause 8).

6.9.1.2 Result

This parameter shall indicate whether the M-ASSOCIATE service request was accepted or rejected.

6.9.1.2.1 Responding Application Reference

This parameter identifies the system that is responding to the M-ASSOCIATE service request. It will normally be identical to the Called Application Reference of the service request, but it need not be so.

6.9.1.2.2 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.1.2.3 Authentication Value

This optional parameter may be included if the calling system requires authentication to allow some services to be performed. It is used to establish the right of the called system to make use of those services.

6.9.1.2.4 User Data

This parameter contains the MMSPdu as formed by the Initiate service response, (see clause 8).

6.9.2 M-RELEASE

The M-RELEASE service allows an MMS-user to terminate an association in an orderly manner. M-RELEASE is a confirmed service. In addition to its other functionality, M-RELEASE allows one MMS-user to transmit some user data (the service parameters of the MMS Conclude service) to its peer MMS-user. Upon completion of the service either (1) the association has been terminated and the MMS environment has been exited (positive result parameter), or (2) the association could not be terminated (negative result parameter). Figure 3 shows the time sequence of service primitives for this service.

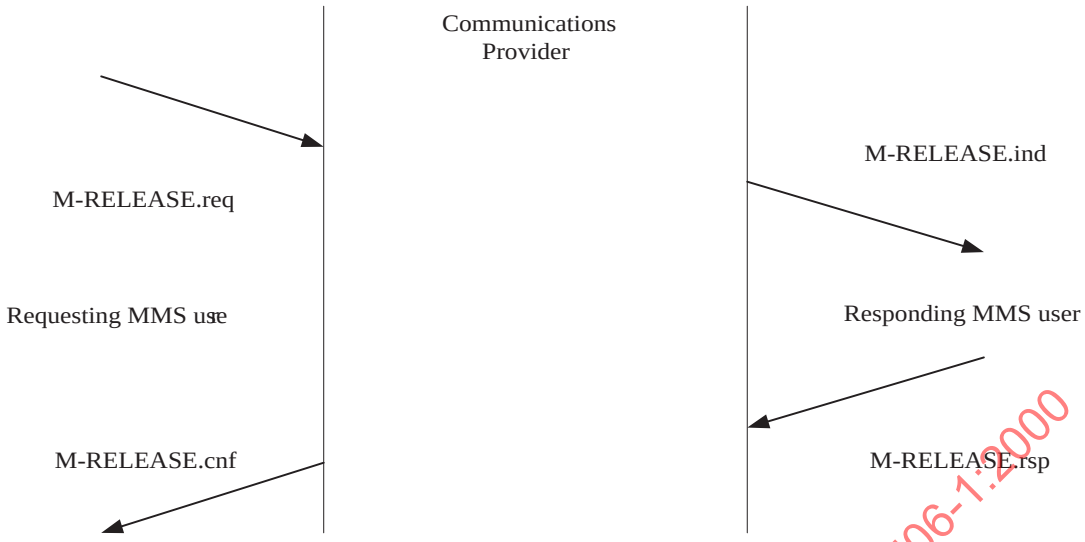

Figure 3 - M-RELEASE service

Table 2 shows the parameters of the M-RELEASE service.

Table 2 - M-Release service parameters

Parameters	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Other Communication Parameters	M	M		
User Data	M	M(=)		
Result			M	M(=)
Other Communication Parameters			M	M
User Data			M	M(=)

6.9.2.1 Argument

This parameter contains the parameters of the M-RELEASE service of the requesting system.

6.9.2.1.1 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.2.1.2 User Data

This parameter contains the MMSPdu as formed by the Conclude service request (see clause 8).

6.9.2.2 Result

This parameter shall indicate whether the M-RELEASE service request was accepted or rejected.

6.9.2.2.1 Other Communication Parameters

This parameter is one or more parameters specific to the underlying communication system being used.

6.9.2.2.2 User Data

This parameter contains the MMSPdu as formed by the Conclude service response, (see clause 8).

6.9.3 M-DATA

The M-DATA allows an MMS-user to transfer an arbitrary amount of data to its peer MMS-user. Most MMS services make use of this service. M-DATA is an unconfirmed service. The principal functionality of M-DATA is to allow one MMS-user to transmit some user data (the service parameters of the MMS confirmed and unconfirmed services) to its peer MMS-user. Figure 4 shows the time sequence of service primitives for this service.

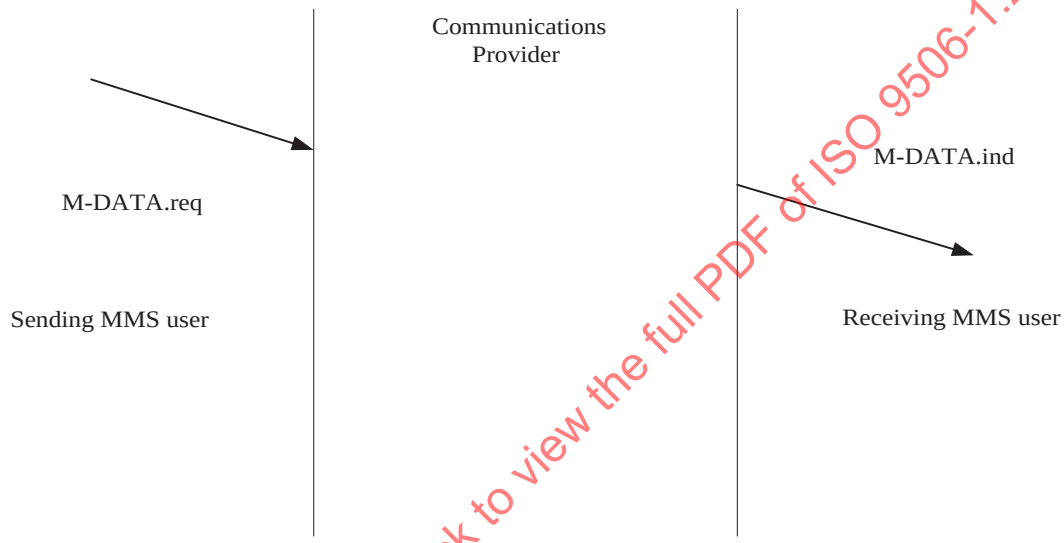

Figure 4 - M-DATA service

Table 3 shows the parameters of the M-DATA service.

Table 3 - M-Data service parameters

Parameters	Req	Ind
Argument	M	M
User Data	M	M(=)

6.9.3.1 Argument

This parameter contains the parameters of the M-DATA service of the requesting system.

6.9.3.1.1 User Data

This parameter contains the MMSPdu as formed by the MMS service request or response.

6.9.4 M-U-ABORT

The M-U-ABORT service allows one MMS-user to abruptly terminate the association. The M-U-ABORT service carries no user data. Figure 5 shows the time sequence of service primitives for this service.

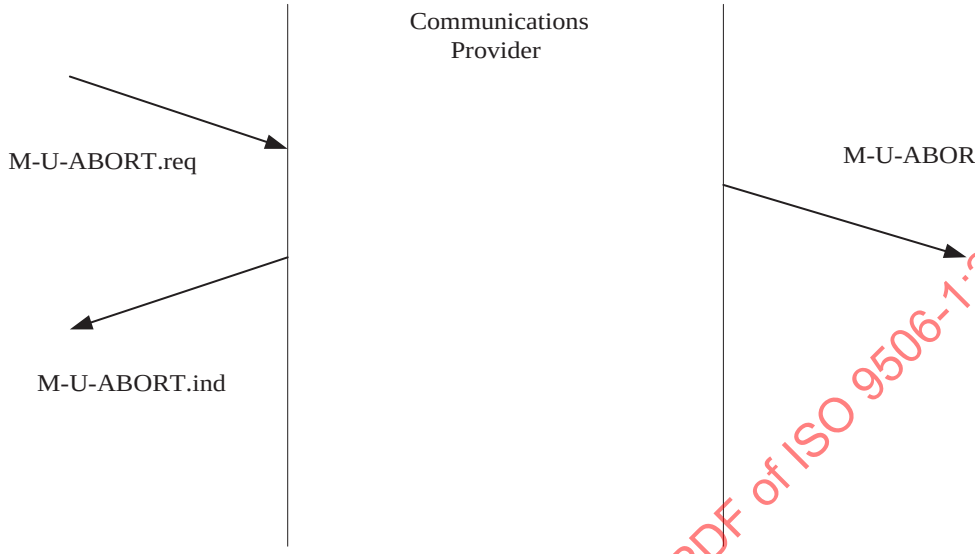

Figure 5 - M-U-ABORT service

Table 4 shows the parameters of the M-U-Abort service.

Table 4 - M-U-Abort service parameters

Parameters	Req	Ind
Argument	M	M
Abort Source		M

6.9.4.1 Argument

This parameter contains the parameters of the M-U-ABORT service of the requesting system.

6.9.4.1.1 Abort Source

This parameter is present in the indication and indicates the source of the abort request. If true, the abort was requested from the local system; if false, the abort was requested from the remote system.

6.9.5 M-P-ABORT

The M-P-ABORT service provides indication to the MMS-user that the communication system has abruptly terminated the association. The M-P-ABORT service carries no user data. Figure 6 shows the time sequence of service primitives for this service.

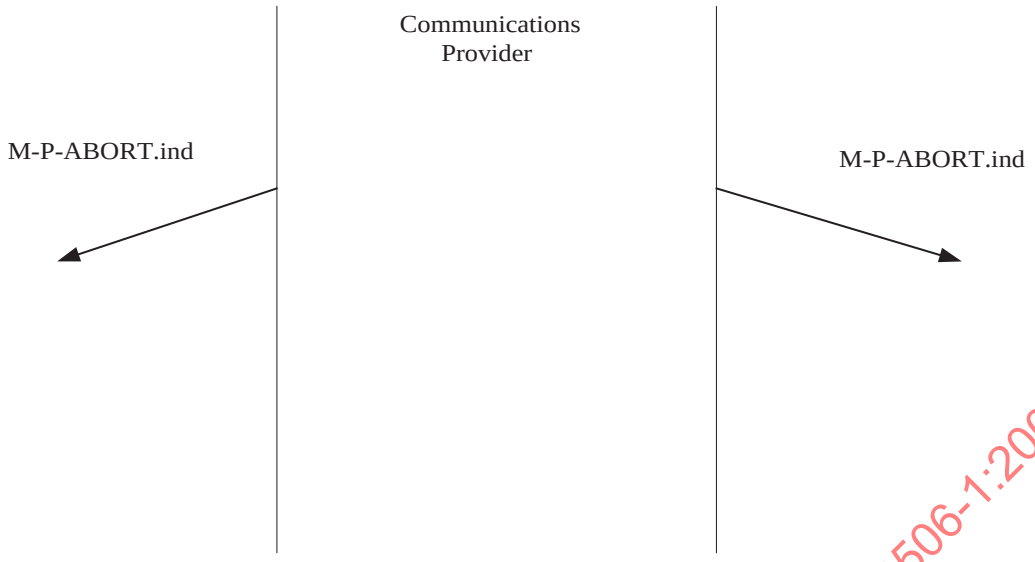

Figure 6 - M-P-ABORT service

Table 5 shows the parameters of the M-P-ABORT service.

Table 5 - M-P-Abort service parameters

Parameters	Req	Ind
Abort Source		M

6.9.5.1 Abort Source

This parameter is present in the indication and indicates the source of the abort request. If true, the abort was requested from the local system; if false, the abort was requested from the remote system.

7 The Virtual Manufacturing Device

7.1 Introduction

This clause provides object models for the following objects:

VMD
NULL-OBJECT
TRANSACTION

7.1.1 The VMD and the real manufacturing device

The MMS services define the externally visible behaviour of an MMS server application process. This behaviour is modelled by describing an entity called a Virtual Manufacturing Device (VMD). This clause provides a model for a VMD. It explains the VMDs relationship to the underlying communication system. It defines the structural elements of the VMD and introduces the abstract objects that exist at a VMD, and are manipulated on behalf of a MMS client.

An implementation of an MMS server provides a mapping of the VMD model on to the functionality of a real manufacturing device. Guidance in the selection of a particular mapping may be found in various Companion Standards. These Companion

Standards address the specific needs of discrete parts manufacturing systems - numerical controllers, programmable controllers, robotic controllers, and vision systems - and of batch and continuous process control systems.

NOTE The MMS services do not constrain the behaviour of a client MMS application process, except with respect to valid sequences of primitives. Therefore a model of the MMS client application process is not provided.

7.1.2 Relationship of the VMD to the OSI Model

A VMD exists within the MMS server application process. It constitutes that portion of an information processing task that makes available - for control, or monitoring, or both - a set of resources and functionality associated with a real manufacturing device. An application process may contain zero or more VMDs. If it does not define a VMD it may not act as an MMS-server.

Each VMD represents a virtual manufacturing device within that AP, and each VMD is logically separate from all other VMDs.

Example: An MMS system that is connected to a non-MMS environment containing multiple attached manufacturing devices could be modelled as a single application process containing one VMD for each attached device or as several application processes, each containing a single VMD for a single, distinct, attached device. The clients of the VMDs in either case will see a particular attached device as a single VMD. Relative to MMS services, this VMD will appear to be independent of all other VMDs.

Each AP may contain zero or more AEs, such that an AE represents a set of communication capabilities of the AP (an AP that contains no AEs may not communicate in the OSI environment).

A VMD may wholly contain zero or more AEs. Each AE in a VMD represents a set of communication capabilities used by the aspects of the AP represented by a VMD. Each AE is related to one and only one VMD. If a VMD contains more than one AE, it contains more than one set of communication capabilities.

7.1.3 Relationship of the VMD to a Real Manufacturing Device

A VMD is an abstract representation of a specific set of resources and functionality at the real manufacturing device and a mapping of this abstract representation to the physical and functional aspects of the real manufacturing device. This mapping of a virtual resource to the underlying actual resource is of relatively long duration.

Generally, the resources of a given VMD are distinct from, and independent of, the resources of all other VMDs. When a virtual resource of two (or more) VMDs is mapped to the same underlying physical resource, a mechanism must be provided by the application process(es), and made available through the VMDs, so that clients of the various VMDs may coordinate their access to the single real resource. This coordination may be modelled by requiring each VMD to obtain control of a virtual semaphore whenever disruptive access to the virtual resource is being requested by an MMS service. This virtual semaphore would then be mapped on to a real semaphore that controls access to the real resource. With this approach, the virtual semaphores of the various VMDs are independent when viewed using MMS services. In other words, users of the virtual semaphore of one VMD appear as though part of the resident system when viewed by users of the semaphores of the other VMDs. (Semaphores are described in clause 16.)

MMS describes the operation of a VMD by describing the abstract objects that are manipulated by it, and by describing the set of operations that may be performed on these objects through use of the MMS services.

7.2 The Structure of a VMD

Each VMD contains exactly one Executive Function and zero or more Program Invocations, each of which depend on one or more Domains. The Domain represents a specific use of a set of capabilities of the VMD. The state of the VMD, in the complete sense, is determined by the values of all the attributes of the VMD, including all the attributes of its Domains and their subordinate objects. The elements of the VMD are described using an ASN.1 module containing Object Class definitions for each object. The module begins with an ASN.1 module header exporting symbols to the module contained in ISO 9506-2 that defines the abstract syntax.

```
MMS-Object-Module-1 { iso standard 9506 part(1) mms-object-model-version1(2) }
DEFINITIONS ::= BEGIN

--
-- This ASN.1 specification has been checked for conformance with the
-- ASN.1 standard by the OSS ASN.1 Tools.
--

EXPORTS AccessCondition,
      AdditionalCBBOptions,
      AdditionalSupportOptions,
      Address,
      AlarmAckRule,
      Control-State,
      DomainState,
      EC-State,
      EC-Class,
      EE-Duration,
      EE-Class,
      EventTime,
      Journal-Variable,
      LogicalStatus,
      Modifier,
      normalPriority,
      normalSeverity,
      ParameterSupportOptions,
      PhysicalStatus,
      Priority,
      ProgramInvocationState,
      Running-Mode,
      ServiceSupportOptions,
      Severity,
      Transitions,
      TypeDescription,
      ULState,
      VMDState;

IMPORTS ApplicationReference,
      Authentication-value FROM
      MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1(4) }
      AlternateAccess,
      ConfirmedServiceRequest,
      AttachToSemaphore,
      AttachToEventCondition,
      Data,
      EE-State,
      Identifier,
      Integer8,
      Integer32,
      MMSString,
      ObjectName,
      TimeOfDay,
      TypeSpecification,
      Unsigned32,
      Unsigned8 FROM
      ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) };

 -- Part 1 - Object Model Definitions
--
-- Note - ASN.1 rules for interpreting the object formalism.
--
-- Each field has a field identifier possibly followed by a name.
```

```
-- The field identifier begins with an '&' and is followed by a reference name,
-- beginning with either a lower case or an upper case letter.
--
-- If the field identifier begins with '&' Upper case letter:
--
--    If there is no following name,
--          the field identifies a type.
--    If the following name is mixed case
--          and begins with an upper case letter,
--          or if the following name is upper case
--          and the name of a Universal type,
--          the field identifies a value set.
--    If the following name is upper case
--          and the name of an Object Class,
--          the field identifies an Object Set.
--
-- If the field identifier begins with '&' lower case letter:
--
--    If the following name is upper case
--          and the name of an Object Class,
--          the field identifies an Object (instance).
--    If the following name is mixed case
--          and begins with an upper case letter,
--          or if the following name is upper case
--          and is the name of a Universal type,
--          the field identifies a value.
--
```

7.2.1 VMD Object Model

This clause introduces the model for the VMD.

```
 VMD ::= CLASS {
      &executiveFunction          ApplicationReference,
      &vendorName                 MMSString,
      &modelName                  MMSString,
      &revision                   MMSString,
      &AbstractSyntaxes           ABSTRACT-SYNTAX,
      &EATransactions             TRANSACTION,
      &Associations               APPLICATION-ASSOCIATION,
      &accessControl              ACCESS-CONTROL-LIST,
      &logicalStatus              LogicalStatus,
      &Capabilities               MMSString,
      &physicalStatus             PhysicalStatus,
      &local-detail               BIT STRING,
      &AccessControlLists         ACCESS-CONTROL-LIST,
      &Domains                    DOMAIN,
      &ProgramInvocations         PROGRAM-INVOCATION,
      &UnitControls               UNIT-CONTROL,
IF (vadr)
      &UnnamedVariables           UNNAMED-VARIABLE,
ENDIF
IF (vnam)
      &NamedVariables             NAMED-VARIABLE,
IF (vlis)
      &NamedVariableLists         NAMED-VARIABLE-LIST,
ENDIF
      &NamedTypes                 NAMED-TYPE,
ENDIF
      &DataExchanges              DATA-EXCHANGE,
      &Semaphores                 SEMAPHORE,
      &OperatorStations           OPERATOR-STATION,
      &EventConditions            EVENT-CONDITION,
```

```
    &EventActions                   EVENT-ACTION,
    &EventEnrollments               EVENT-ENROLLMENT,
IF (cspi)
    &EventConditionLists            EVENT-CONDITION-LIST,
ENDIF
    &Journals                       JOURNAL,
IF (csr)
    &operationState                 VMDState,
    &safety-Interlocks-Violated     BOOLEAN,
    &any-Resource-Power-On          BOOLEAN,
    &all-Resources-Calibrated       BOOLEAN,
    &local-Control                  BOOLEAN,
    &selected-Program-Invocation    PROGRAM-INVOCATION
ENDIF
    }
```

7.2.1.1 &executiveFunction

The &executiveFunction field identifies the VMD. The value of this field is of type ApplicationReference. For OSI specific communications systems, this type is defined in Annex A. For other communication systems, a similar definition of ApplicationReference must be supplied. The existence of a fully functional &executiveFunction exactly corresponds with the existence of the VMD.

7.2.1.2 &vendorName

The &vendorName field is a character string that identifies the vendor of the system that supports this VMD.

7.2.1.3 &modelName

The &modelName field is a character string that identifies the model of the system that supports this VMD. The value of this string is normally assigned by the vendor.

7.2.1.4 &revision

The &revision field is a character string that identifies the revision level of the system that supports this VMD. The value of this string is normally assigned by the vendor.

7.2.1.5 &AbstractSyntaxes

The &AbstractSyntaxes field identifies the set of Abstract Syntaxes that this VMD is able to support in the MMS Application Context. This field includes any abstract syntax that can be recognized as an encoding for Load Data and Execution Argument. The abstract syntaxes defined in ISO 9506-2 shall not be included in the field. The ABSTRACT-SYNTAX object is defined in ISO/IEC 8824-2, Annex B.

7.2.1.6 &EATransactions

This field identifies those transaction objects that do not explicitly depend on an application association. Such transactions occur through the processing of Event Actions. Transaction objects are described in 7.4.1.

7.2.1.7 &Associations

This field identifies the associations established between this VMD and external MMS clients. In order for MMS services to be used, at least one such association is required. Association establishment and the Application Association object are described in clause 8.

7.2.1.8 &accessControl

This field identifies an Access Control List object that specifies necessary (but not sufficient) conditions for an MMS service to succeed. The conditions specified in this Access Control List object shall be satisfied for the service class corresponding to the requested service in order for the service to succeed. Additional conditions for success may be imposed by an Access Control List object referenced by the object of the service request. If no other specification has been provided, this field should reference the predefined symbol 'M_NonDeletable'. The Access Control List object is described in clause 9.

7.2.1.9 &logicalStatus

The &logicalStatus field identifies one of four levels of functionality of the VMD available through MMS.

```
LogicalStatus ::= [0] IMPLICIT INTEGER {
    state-changes-allowed          (0),
    no-state-changes-allowed       (1),
    limited-services-permitted     (2),
    support-services-allowed       (3) }
```

7.2.1.9.1 state-changes-allowed

If &logicalStatus has this value, all MMS services supported by this VMD may be performed.

7.2.1.9.2 no-state-changes-allowed

If &logicalStatus has this value, the only MMS services that may be performed (if the VMD supports the service) are:

Abort
Conclude
Cancel
GetAccessControlListAttributes
GetAlarmEnrollmentSummary
GetAlarmSummary
GetCapabilityList
GetDataExchangeAttributes
GetDomainAttributes
GetECLAttributes
GetEventActionAttributes
GetEventConditionAttributes
GetEventEnrollmentAttributes
GetNamedTypeAttributes
GetNamedVariableListAttributes
GetNameList
GetProgramInvocationAttributes
GetUnitControlAttributes
GetVariableAccessAttributes
Identify
Initiate
Read
ReadJournal
ReportAccessControlledObjects
ReportECLStatus
ReportEventActionStatus
ReportEventConditionStatus
ReportEventEnrollmentStatus
ReportJournalStatus
ReportPoolSemaphoreStatus
ReportSemaphoreEntryStatus
ReportSemaphoreStatus
Status

7.2.1.9.3 limited-services-permitted

If &logicalStatus has this value, the only MMS services that may be performed are: Abort, Conclude, Status and Identify.

7.2.1.9.4 support services allowed

If &logicalStatus has this value, all MMS services supported by the VMD except the Start, Stop, Reset, Resume, Kill, StartUnitControl, and StopUnitControl services may be performed.

7.2.1.10 &Capabilities

A VMD represents the real device to the MMS client by providing capabilities that may be used by the MMS client in order to effect some control or monitoring (or both) activity through the VMD.

A capability is a locally defined resource (physical or logical) or a locally defined set of resources that can be identified by a character string. The definition and management of capabilities is outside the scope of this part of ISO 9506. However, it is assumed that the existence and status of capabilities is known by the &executiveFunction, and that the &executiveFunction

contains sufficient knowledge in order to "allocate" capabilities to the various Domains that may be created. No object model is provided for a capability, since it is considered primitive from the MMS point of view.

A capability may be sharable (or not) depending on local criteria. Additionally, a capability may be distinct, or it may encompass (or be encompassed by) one or more other capabilities.

NOTE An implementation of an MMS server should provide a mapping of the VMD model on to the functionality of a real device. Guidance in the selection of a particular mapping may be found in various Companion Standards. In particular, the Companion Standards will aid in the identification of Domains and Program Invocations with features of real systems. The field "capability" is intended to allow expression of features of the implementation needed for relating any real implementation to the VMD model and to the extended model provided by the Companion Standard. The use of capabilities in any MMS implementation requires prior agreement between implementations of MMS client and server.

7.2.1.11 &physicalStatus

Associated with each capability is one or more attributes that describe the state of the capability. It is outside the scope of this part of ISO 9506 to provide a uniform representation of these attributes. However, it is useful to provide a standard representation of the gross aspects of all the capabilities, taken together, in order to characterize the operational state of the hardware. The &physicalStatus field represents this attribute (of the entire collection of capabilities) of the real device.

NOTE This status refers to the hardware associated with the device's operation. It is unrelated to the ability of the device to communicate.

```
PhysicalStatus ::=      [1] IMPLICIT INTEGER {
    operational                 (0),
    partially-operational       (1),
    inoperable                  (2),
    needs-commissioning         (3) }
```

7.2.1.11.1 operational

The real device associated with this VMD has no known deficiencies and is able to perform its intended tasks.

7.2.1.11.2 partially-operational

One or more functions of the real device cannot be performed due to hardware malfunctions or limitations.

7.2.1.11.3 inoperable

One or more significant problems exist at the real device that prevent it from doing any useful task.

7.2.1.11.4 needs-commissioning

The device is in a state such that a local commissioning process needs to be performed before useful tasks can be accomplished.

7.2.1.12 &local-detail

This attribute is a bitstring that represents additional detail about the status of the VMD as specified by the vendor of the device. The maximum length of the bitstring is 128 bits. The contents of this attribute is a local matter and not a subject for further standardization.

7.2.1.13 &AccessControlLists

This field identifies the Access Control List objects whose name scope is VMD-specific. Access Control List objects are described in clause 9.

7.2.1.14 &Domains

A Domain represents a specific instance of use of a set of capabilities of the VMD. A Domain includes those aspects of a VMD that are associated with a specific element (possibly all) of a coordinated control or monitoring (or both) strategy. The allocation of the capabilities of a VMD to the Domains may be static or it may be dynamic. If the allocation is static, the Domain is predefined within the MMS server and its name is known. If the allocation is dynamic, the Domain comes into existence and is removed from the VMD either through the action of MMS services or through local actions. Within a given VMD, either or both types of Domains may exist. Domains are further described and an object model is provided in clause 11.

A Domain may be empty or it may contain "information". The "information" may be program instructions for some processor or tables of values or other classes of data or all of the above. For dynamic Domains that come into existence through MMS services, the Domains are implicitly created by the process of loading their contents. Dynamic Domains may also be created through actions of the Program Invocation when it is in execution or through other local means.

The Domain content is often closely associated with a file. The concept of file, although it appears in some MMS services, is not defined within MMS. The file may be part of a virtual Filestore (ISO 8571) that is part of the Application Process that contains the VMD, or it may be locally defined. If the "information" within a file is a part of the VMD and can be used directly by the VMD (as a component of a Program Invocation), it can be considered to be a Domain. The file, however, must be considered a separate object, outside the scope of the VMD.

Many MMS objects may be defined subordinate to (within the name space of) a Domain. Thus, a Domain represents a single name space in which MMS objects are uniquely identifiable.

7.2.1.15 &ProgramInvocations

A Program Invocation consists of a set of procedural and data elements contained within Domains together with execution control information. These elements may be predefined within the VMD, or they may be dynamically defined (for example by creation of a Domain through use of the MMS load services), or both. The Program Invocation itself may be predefined within the VMD, or it may be dynamically created and deleted either by local means or through the use of MMS services. An object model of the Program Invocation is given in clause 12.

7.2.1.16 &UnitControls

This field identifies the Unit Control objects in the VMD. Unit Controls always have a name scope that is VMD-specific. Unit Control objects are described in clause 13.

7.2.1.17 &UnnamedVariables

The VMD may allow direct access to the memory space of the underlying computer. This field is present only if the **vadr** parameter CBB has been negotiated. Such access is characterized by a machine address and possibly a type specification corresponding to that address. Unnamed Variables are described in clause 14.

7.2.1.18 &NamedVariables

This field identifies the Named Variable objects whose name scope is VMD-specific. This field is present only if the **vnam** parameter CBB has been negotiated. Named Variables are described in clause 14.

7.2.1.19 &NamedVariableLists

This field identifies the Named Variable List objects whose name scope is VMD-specific. This field is present only if the **vnam** and the **vlis** parameter CBB's have been negotiated. Named Variables Lists are described in clause 14.

7.2.1.20 &NamedTypes

This field identifies the Named Type objects whose name scope is VMD-specific. This field is present only if the **vnam** parameter CBB has been negotiated. Named Types are described in clause 14.

7.2.1.21 &DataExchanges

This field identifies the Data Exchange objects whose name scope is VMD-specific. Data Exchange objects are described in clause 20.

7.2.1.22 &Semaphores

This field identifies the Semaphore objects whose name scope is VMD-specific. Semaphores are described in clause 16.

7.2.1.23 &OperatorStations

This field identifies the Operator Station objects whose name scope is VMD-specific. Operator Stations are described in clause 17.

7.2.1.24 &EventConditions

This field identifies the Event Condition objects whose name scope is VMD-specific. Event Conditions are described in clause 19 and event processing is described in clause 18.

7.2.1.25 &EventActions

This field identifies the Event Action objects whose name scope is VMD-specific. Event Actions are described in clause 20 and event processing is described in clause 18.

7.2.1.26 &EventEnrollments

This field identifies the Event Enrollment objects whose name scope is VMD-specific. Event Enrollments are described in clause 21 and event processing is described in clause 18.

7.2.1.27 &EventConditionLists

This field identifies the Event Condition List objects whose name scope is VMD-specific. Event Condition Lists are described in clause 22 and event processing is described in clause 18.

7.2.1.28 &Journals

This field identifies the Journal objects whose name scope is VMD-specific. Journals are described in clause 16.

7.2.1.29 &operationState

This field represents the state of the system. This field is present only if the **csr** parameter CBB has been negotiated. The field is used solely for systems (such as robots) in which the VMD represents a single, complex piece of machinery with operational states. The relationship of this field to the physical devices and to the state of the Controlling Program Invocations is defined below.

```
VMDState ::= INTEGER {
    idle                        (0),
    loaded                      (1),
    ready                       (2),
    executing                   (3),
    motion-paused               (4),
    manualInterventionRequired  (5) }
```

7.2.1.30 &safety-Interlocks-Violated

This field, of type boolean, indicates the state of the safety interlocks. This field is present only if the **csr** parameter CBB has been negotiated. The field is used solely for robots and similar equipment having a safety interlock mechanism. If the value of

this field is true, the safety interlocks have been violated since the system was last reset. The method of resetting the safety interlocks is a local matter.

7.2.1.31 &any-Resource-Power-On

This field, of type boolean, indicates whether (true) or not (false) any physical resource has power applied to it. This field is present only if the **csr** parameter CBB has been negotiated. The value of this field is the logical 'OR' of a similar primitive attribute of each physical resource in the system.

7.2.1.32 &all-Resources-Calibrated

This field, of type boolean, indicates whether (true) or not (false) all the physical resources in the system that have calibration attributes are in fact calibrated. This field is present only if the **csr** parameter CBB has been negotiated.

7.2.1.33 &local-Control

This field, of type boolean, indicates whether (true) or not (false) some local agent has control of any of the physical resources of the system. This field is present only if the **csr** parameter CBB has been negotiated. Having control indicates the ability to cause actions that change the physical resources and hence the attributes that represent those resources. &local-Control can r eflect either a human operator or some automatic procedure, either of which inhibit the assertion of control remotely.

If &local-Control is false, control may reside in some remote agent. This condition can also reflect a condition in which the system is running under no direct control other than its task program.

The value of the &local-Control field, the &operation-State field, and the &logical-Status field are interrelated. Table 6 illustrates the relationship between these attributes.

Table 6 - Local control

Local Control	Operation State	VMD Logical Status
TRUE	Any	NO-STATE-CHANGES-ALLOWED or LIMITED-SERVICES-PERMITTED or SUPPORT-SERVICES-ALLOWED
FALSE	MANUAL-INTERVENTION-REQUIRED	NO-STATE-CHANGES-ALLOWED or LIMITED-SERVICES-PERMITTED or SUPPORT-SERVICES-ALLOWED
	Any other	Any

NOTE While &local-Control is true, the &operation-State can continue to change. However, it is a local matter how the state changes occur.

7.2.1.34 &selected-Program-Invocation

This field identifies the Program Invocation that has been selected to control the operation of the system. This field is present only if the **csr** parameter CBB has been negotiated. This Program Invocation has its &control field equal to **controlling** and has been selected through the use of the Select operation.

7.2.2 Initialization of the VMD

Clause 25 of ISO 9506-2 describes the initialization of the VMD.

7.2.3 Services on the VMD

Clause 10 describes the MMS services that operate on the VMD in its entirety.

7.3 The NULL Object

Since the ASN.1 standard does not allow an object set to be empty, it is necessary to introduce a special object instance, the NULL object. This object is included in every object set in the MMS model. If the object set has no other members than the NULL object, the set is considered to be empty from the MMS point of view. The following defines the null object.

```
NULL-OBJECT ::= CLASS { &null NULL }

null-instance NULL-OBJECT ::= { &null NULL }
```

7.4 Transactions

This clause introduces a model of the Transaction and its processing.

7.4.1 Transaction Object Model

Most MMS services are confirmed services (see ISO 9506-2, clause 6). When the VMD receives an indication primitive for one of the confirmed services, a Transaction object is created that governs the processing of this service. The description of that object follows:

```
TRANSACTION ::= CLASS {
    &invokeID                       INTEGER,
IF ( attachToEventCondition attachToSemaphore )
    &Pre-excutionModifiers          Modifier,
    &currentModifier                CHOICE {
        modifier                        Modifier,
        none                            NULL },
ENDIF
    &confirmedService-Request       ConfirmedServiceRequest,
IF ( attachToSemaphore )
    &Post-executionModifiers        Modifier,
ENDIF
    &cancelable                     BOOLEAN }
```

7.4.1.1 &invokeID

This field is an integer that serves to identify the transaction within the Application Association.

7.4.1.2 &Pre-excutionModifiers

This field is an ordered set of modifiers to be satisfied before execution of the Confirmed Service Request can begin. This set may be empty.

```
Modifier ::= SEQUENCE {
    modifierID                  INTEGER,
    modifier                    CHOICE {
IF ( attachToEventCondition )
        eventModifier               AttachToEventCondition,
ENDIF
IF ( attachToSemaphore )
        semaphoreModifier           AttachToSemaphore
ENDIF
        } }
```


7.4.1.2.1 eventModifier

The AttachToEventCondition modifier is described in clause 18.

7.4.1.2.2 semaphoreModifier

The AttachToSemaphore modifier is described in clause 16.

7.4.1.3 &currentModifier

This field identifies the Semaphore Entry object or the Event Enrollment object controlling the current modifier's execution. This field is initially empty. It changes to identify the modifier in the Set of Pre-Execution Modifiers as they are processed. If the Set of Pre-Execution Modifiers is empty, this field will always be empty.

7.4.1.4 &ConfirmedService-Request

This field identifies the pending service, including its arguments.

7.4.1.5 &Post-executionModifiers

This field identifies the set of Semaphore Entry objects that this service invocation owns due to a processed Attach to Semaphore modifier. Attach to Event Condition modifiers do not occur on the list of &Post-executionModifiers.

7.4.1.6 &cancelable

This field identifies whether or not the service may be cancelled. Initially true, the MMS server may set this field false during the processing of the service request.

7.4.2 Initialization of Transaction objects

A Transaction object shall be created upon receipt of an indication service primitive for an MMS confirmed service, and deleted after the MMS-user issues a response service primitive for that service instance. The number of Transaction objects that may exist at any time is governed by the negotiated maximum number of services outstanding (see 8.2).

Upon receipt of a Confirmed Service indication, the MMS Server shall create a new Transaction object and initialize it as follows:

```
transaction TRANSACTION ::= {
    &invokeID                   newID,
    &Pre-excutionModifiers      serviceParameter,
    &CurrentModifier      none: NULL,
    &confirmedService-Request   serviceParameter,
    &Post-executionModifiers    { null-instance },
    &cancelable                 TRUE
    }
```

NOTE This production is not valid ASN.1 as written. The MMS Server should substitute a valid integer value for newID to identify the transaction, and insert the parameters from the Confirmed Service indication for the &Pre-executionModifiers and the &confirmedService-Request. The other initialization parameters are valid.

7.4.2.1 &invokeID

The MMS server shall assign an integer value to this field. This value shall be unique to this application association.

7.4.2.2 &Pre-excutionModifiers

The MMS server shall assign a value set to this field corresponding to the parameters of the service indication.

7.4.2.3 &CurrentModifier

The MMS server shall assign an initial value of **none** to this field.

7.4.2.4 &confirmedService-Request

The MMS server shall assign a value to this field corresponding to the parameter of the service indication.

7.4.2.5 &Post-executionModifiers

The MMS server shall assign an empty value set as the initial value of this field.

7.4.2.6 &cancelable

The MMS server shall assign the value true as the initial value of this field. The Cancelable attribute of the Transaction object shall not be set to false during the processing of the pre-execution modifiers. After all pre-execution modifiers have been processed, the cancelable attribute may be set to false by the MMS-user at any time as a local matter, subject to the requirements for the Cancel service and the service named in the Confirmed Service Request. The cancelable attribute shall have been set to false by the time that post-execution modifier processing takes place.

7.4.3 Processing of Transaction objects

A Transaction object shall be created either upon receipt of an indication service primitive for an MMS confirmed service or as part of the processing of an event occurrence (see 18.1.1). The Transaction object shall be deleted after the MMS-user issues a response service primitive for that service instance. The number of Transaction objects that may exist at any time is governed by the negotiated maximum number of services outstanding (see 8.2).

NOTE Although this part of ISO 9506 requires that Transaction objects be initialized in the order of receipt, there is no such requirement for the order of processing of these objects. To enforce serialization of actions, the MMS client should wait for a service response before issuing subsequent service requests.

Each of the pre-execution modifiers shall be processed in order. Each modifier shall complete successfully before the next modifier is processed. If a modifier fails (see definition of particular modifiers), the MMS-user shall process the Post-execution modifier list (as specified below) and issue a Response(-) service primitive, specifying the appropriate error class and code, and the Transaction object shall be deleted. The &CurrentModifier field shall be set to indicate the modifier currently being processed in the list. The &CurrentModifier field shall be identified by an integer, where 1 indicates the first element in the list. For each AttachToSemaphore modifier that is successfully processed, an entry shall be made in the &Post-executionModifiers field (in order to allow relinquishing of control of the semaphores after service execution). The &Post-executionModifiers field shall be constructed in reverse order, such that the first pre-execution modifier becomes the last post-execution modifier.

After all pre-execution modifiers have been processed successfully (if any are specified on the indication service primitive), the Confirmed Service Request shall be executed in accordance with the service procedure specified in this part of ISO 9506 for the named service.

Following completion of the service request, the MMS-user shall process the post-modifiers in the List Of &Post-executionModifiers field in the order that they are specified in the list (note that this is the opposite of the order in which the pre-execution modifiers were executed).

After all post-execution modifiers have been processed, the response service primitive shall be issued by the MMS-user. The &CurrentModifier field is used to indicate the current modifier during the processing of this list.

7.5 Specification of Named objects

MMS objects are normally referenced by name. The name of an object shall be unique within its scope of definition and within the class of object it identifies.

7.5.1 Scope of Names

MMS names can have one of three scopes, VMD-specific Domain-specific, and Application Association-specific.

7.5.1.1 VMD-specific Scope

A name that has VMD-specific scope shall be unique among all VMD-specific objects of the same object class. Such a name may be referenced by all clients of the VMD instance on any application association with it. VMD-specific objects persist after the MMS application association ceases to exist.

7.5.1.2 Domain-specific Scope

A Domain represents a single flat name space. A name that is Domain-specific shall be unique for its class of object within the Domain in which it is defined. The unique identification of such an object requires the specification of the name of the Domain and the name of the object, thus implying a two level naming hierarchy.

7.5.1.3 Application Association-specific Scope

A name having "Application Association-specific" (AA-specific) scope may only be referenced by the MMS client for which the name was defined, and only on the specific application association over which the name's definition is valid. The definition of an object bearing an application association-specific name, if not explicitly deleted previously, shall be deleted whenever the defining application association ceases to exist.

7.5.2 Classes of objects

The specific instances of MMS objects that can be named are listed in Table 7. Not all objects can have all possible Name Scopes. The allowed combinations are indicated by Xs in Table 7.

Table 7 - Name Class and Scope

	VMD	Domain	AA	clause
Access Control List Objects	X			9
Domain Objects	X			11
Program Invocation Objects	X			12
Unit Control Objects	X	X	X	13
Named Variable Objects	X	X	X	14
Named Variable List Objects	X	X	X	14
Named Type Objects	X	X	X	14
Data Exchange Objects	X		X	15
Semaphore Objects	X			16
Operator Station Objects	X			17
Event Condition Objects	X	X	X	19
Event Action Objects	X			20
Event Enrollment Objects	X	X	X	21
Event Condition Lists	X	X	X	22
Journal Objects	X		X	23

There are other limitations on the names of Event Conditions, Event Actions, and Event Enrollments. Creation of a semaphore automatically creates an Event Condition of the same name (see 16.4). Therefore, a semaphore cannot be proposed for creation that has the same name as an existing Event Condition. If the monitor parameter is selected, the creation of a Program Invocation automatically creates an Event Condition, and Event Action, and an Event Enrollment, all having the same name as the Program Invocation (see 12.2). Therefore, a Program Invocation cannot be proposed for creation whose name is the same as an existing Event Condition, Event Action, or Event Enrollment.

7.5.3 Object lifetime

Each of the objects listed above has a lifetime within the VMD that can be inferred from the scope of its name.

VMD-specific Scope - Such objects exist as long as the VMD exists (unless explicitly deleted).

Domain-specific Scope - Such objects exist as long as the Domain on which they depend exists (unless explicitly deleted).

AA-specific Scope - Such objects exist only as long as the Application Association over which they were defined continues to exist (unless explicitly deleted).

7.5.4 Object visibility

The Name Scope of a named object also determines the visibility of the object. An object whose name is VMD-specific or Domain-specific may be referenced by any association to the VMD; an object whose name is AA-specific may only be referenced on the association over which the object was defined.

7.5.5 Creation of MMS objects

Each of the MMS objects can either be static, that is predefined within the implementation, or dynamic, coming into existence during the course of operation of the VMD. Static objects usually may not be deleted through the use of MMS services, and dynamic objects usually may be deleted, but there may be exceptions to either rule. Static objects are predefined by the implementation and have names assigned to them either by the implementor, or in conformance to this part of ISO 9506 or to one of the Companion Standards.

Dynamic objects can come into existence either (1) through explicit MMS service procedures, (2) through local action of the system or of the system operators, or (3) through the execution of some Program Invocation. The means of local creation is outside the scope of this part of ISO 9506. However, if a locally created object is to be visible to MMS services, its external behaviour, including all its visible attributes, shall be consistent with this service definition.

7.5.6 Deletion of MMS objects

All MMS objects subordinate to the VMD may be deleted from the VMD through appropriate MMS service requests if such requests are permitted (see 9.1.1.8). In addition to this explicit method of deletion, MMS objects may also be deleted through local action of the system or of the system operators or through the execution of some Program Invocation. The means of local deletion is outside the scope of this part of ISO 9506. Objects that are subordinate to a Domain are automatically deleted when the Domain is deleted. This is true regardless of any conditions specified in the Access Control List object referenced by the object subordinate to the Domain.

7.5.7 Alteration of MMS objects

All MMS objects have as part of their description a list of externally visible attributes. In addition to the MMS services that alter these attributes, these attributes may also be changed through the local action of the system or of the system operators, or through the execution of a Program Invocation. In particular, the values of variables associated with Domains or with the VMD directly may change due to the execution of a Program Invocation. The means of local alteration of object attributes is outside the scope of this part of ISO 9506.

A VMD should, if possible, guarantee that access to any MMS object required by an MMS service is not interruptible. In other words, for the entire duration of the MMS service, the object should have a set of attributes that represent the state of the VMD at a single instant of time. Since such guarantees may not be possible, or if possible for some objects may not be possible for all objects, the static conformance statement for the VMD shall state whether uninterruptible access is supported and, if supported, under what constraints it is guaranteed.

7.5.8 Control of Access to MMS objects

This International Standard provides explicit control for the ability to access or alter MMS named objects. Each named object within an MMS implementation contains a reference to an Access Control List object that specifies the conditions under which services directed at the named object may succeed. For the purposes of specifying the control conditions, services are grouped into seven classes, read, write, load, store, execute, delete, and edit. The control conditions include possession of a semaphore, identity of user (Application Reference), and the submission of a password (which may be arbitrarily complex). These conditions are necessary but not sufficient for the success of the service. If the conditions are not satisfied, the service is required to fail; the service may always fail for reasons beyond the scope of this standard. These conditions may be combined in arbitrary ways. Conditions may be specified separately for individual objects and for all objects of the VMD. Conditions restricting creation of objects can only be specified for the entire VMD.

In the second amendment to the first edition of MMS, the &accessControlList field of named objects replaced the MMS Deletable attribute of the first edition of MMS. For backward compatibility, a derivation rule from the &accessControlList is provided for services that report the value of the MMS Deletable attribute. Using this rule, implementations of earlier versions of MMS will be able to interwork with implementations of this version of MMS as long as the additional services specified in this version are not employed.

The **aco** parameter CBB is used to indicate whether or not the object reporting services shall report attributes related to the use of Access Control Lists.

7.6 Object Name structure

The parameter Object Name occurs frequently in the specification of MMS services. The structure of the Object Name is shown in Table 8.

Table 8 - Object Name

Object Name	Req/Rsp	Ind/Cnf
Name Scope	M	M(=)
AA-specific	S	S(=)
Item Identifier	M	M(=)
Domain-specific	S	S(=)
Domain Identifier	M	M(=)
Item Identifier	M	M(=)
VMD-specific	S	S(=)
Item Identifier	M	M(=)

7.6.1 Name Scope

This parameter specifies which of the possible scopes is to be used for this Object Name.

7.6.2 AA-specific

Some named objects may have an AA-specific name. AA-specific objects shall only exist as a result of a definition of the name over the association by an MMS Service. As there is no MMS service to create AA-specific Data Exchange objects, these objects may be created by local action at the time of association establishment.

7.6.2.1 Item Identifier

This parameter, of type Identifier, is the name of the object. This parameter shall be unique within this application association for the class of object named.

7.6.3 Domain-specific

Named objects that depend on a Domain may have names that are Domain-specific.

7.6.3.1 Domain Identifier

This parameter, of type Identifier, is itself a VMD-specific name of the Domain that contains the named object.

7.6.3.2 Item Identifier

This parameter, of type Identifier, is the name of the object within the named Domain. This parameter shall be unique within the named Domain for the class of object named.

7.6.4 VMD-specific

Any named object may have a VMD-specific name.

7.6.4.1 Item Identifier

This parameter, of type Identifier, is the name of the object. This parameter shall be unique within the VMD for the class of object named.

7.7 Object Class structure

The parameter Object Class occurs frequently in the specification of MMS services. The structure of the Object Class is shown in Table 9.

Table 9 - Object Class

Object Class	Req/Rsp	Ind/Cnf
Object Class	M	M(=)
Basic Object Class	S	S(=)
Extended Object Class	S	S(=)

7.7.1 Object Class

This parameter specifies the class of object under consideration. One of the following parameters shall be chosen.

7.7.2 Basic Object Class

This parameter shall specify the class of the object to be renamed. It shall specify a choice of the following objects:

a) Named Variable
b) Scattered Access
c) Named Variable List
d) Named Type
e) Semaphore
f) Event Condition
g) Event Action
h) Event Enrollment
i) Journal
j) Domain
k) Program Invocation
l) Operator Station
m) Data Exchange
n) Access Control List

7.7.3 Extended Object Class

This parameter shall specify the class of the object to be renamed. It shall specify a choice of the following objects:

a) EventConditionList
b) UnitControl

8 Environment And General Management services

8.1 Introduction and Models

This clause provides a model for the following object:

APPLICATION-ASSOCIATION

This clause specifies the following services:

Initiate
Conclude
Abort
Cancel
Reject

8.1.1 Environment Management State Diagram

This clause defines the state diagram for entering and leaving the MMS environment. The initial state for both the calling and called MMS users shall be the state "No MMS Environment". The state diagram is depicted from the point of view of an MMS-user. The state of the MMS environment is within the context of a specific application association. Restrictions on the use of MMS services within this environment shall not apply to other environments established by the MMS-user on different application associations.

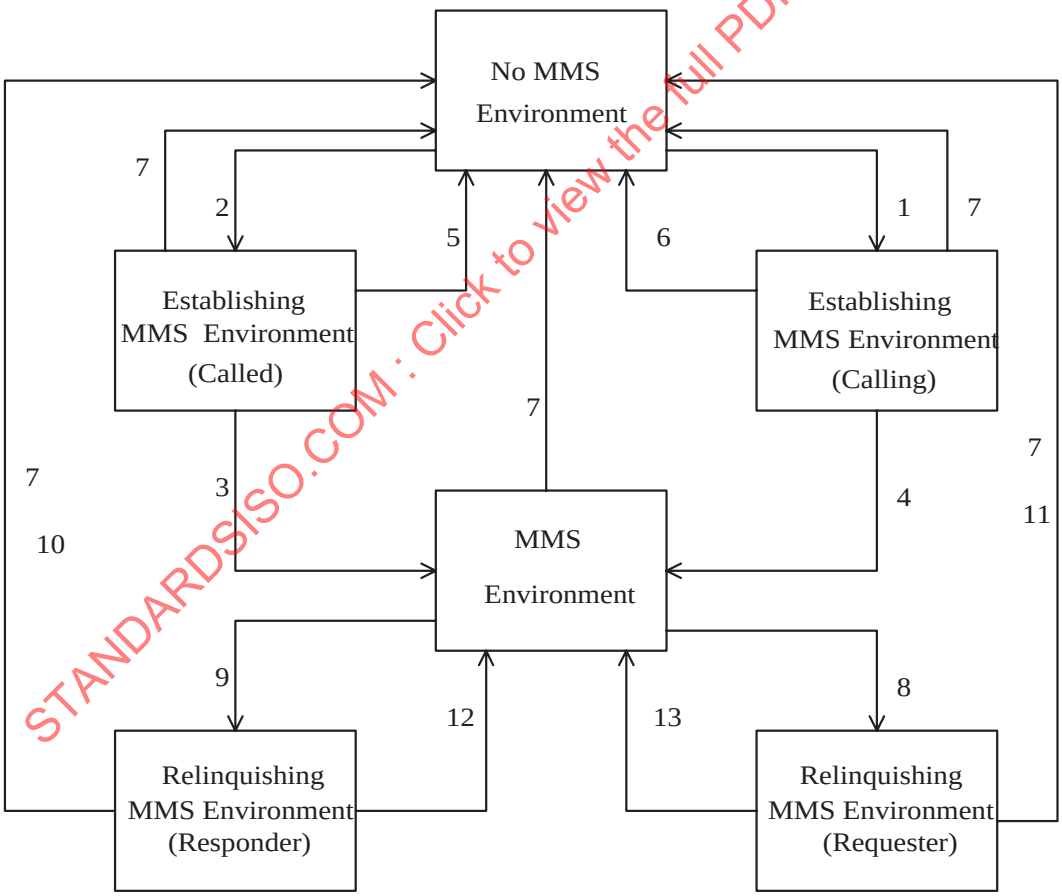

Figure 7 - Environment Management State Diagram

Transitions:
1 - initiate.request
2 - initiate.indication
3 - initiate.response(+)
4 - initiate.confirm(+)
5 - initiate.response(-)
6 - initiate.confirm(-
7 - abort.request or abort.indication

8 - conclude.request
9 - conclude.indication
10 - conclude.response(+)
11 -conclude.confirm(+)
12 - conclude.response(-)
13 - conclude.confirm(-)

8.1.2 Restriction on Use of MMS services

This clause places restrictions on the use of MMS services for certain states. The restrictions in this clause are supplementary to other restrictions placed by other clauses in this part of ISO 9506.

8.1.2.1 The "No MMS Environment" state

While in the state "No MMS Environment", an MMS-user shall only issue the initiate.request service primitive.

8.1.2.2 The "Establishing MMS Environment (Calling) state

While in the state "Establishing MMS Environment (Calling)", an MMS-user shall only issue the abort.request service primitive.

8.1.2.3 The "Establishing MMS Environment (Called) state

While in the state "Establishing MMS Environment (Called)", an MMS-user shall only issue the initiate.response or abort.request service primitives.

8.1.2.4 The "MMS Environment" state

Clauses 8 to 23 place requirements on the use of service primitives in the state "MMS Environment". This state is divided into a series of substates, which are described in those clauses. All MMS states except the "No MMS Environment", "Establishing MMS Environment (Calling)", "Establishing MMS Environment (Called)", "Relinquishing MMS Environment (Requester)" and "Relinquishing MMS Environment (Responder)" are substates of the "MMS Environment" state.

The only events that cause an exit from the "MMS Environment" state are the issuance of an abort.request service primitive, the issuance of a conclude.request service primitive, the receipt of an abort.indication service primitive, or the receipt of a conclude.indication service primitive.

While in the state "MMS Environment", an MMS-user may issue any request or response service primitive, subject to the following restrictions:

a) other clauses of this part of ISO 9506 restrict the usage of services and service primitives via sequencing requirements;

b) a response primitive shall not be issued unless a request primitive for a corresponding service indication has been received (see clause 7);

c) the initiate.request service primitive shall not be issued.

8.1.2.5 The "Relinquishing MMS Environment (Requester)" state

While in the state "Relinquishing MMS Environment (Requester)", the only request primitive that the MMS-user may issue is the abort.request service primitive. Note that responses, either (+) or (-) may continue to be issued.

8.1.2.6 The "Relinquishing MMS Environment (Responder)" state

While in the state "Relinquishing MMS Environment (Responder)", the MMS-user may only issue the abort.request and conclude.response service primitives.

8.1.3 Application Association

This clause introduces the model of the application association.

```
APPLICATION-ASSOCIATION ::= CLASS {
    &aaIdentifier               INTEGER,
    &client                     ApplicationReference,
    &abstractSyntax             ABSTRACT-SYNTAX,
    &authenticationValue        Authentication-value OPTIONAL,
-- This field represents a 'user password'
    &Transactions               TRANSACTION,
IF (vnam)
    &NamedVariables             NAMED-VARIABLE,
IF (vlis)
    &NamedVariableLists         NAMED-VARIABLE-LIST,
ENDIF
    &NamedTypes                 NAMED-TYPE,
ENDIF
    &DataExchanges              DATA-EXCHANGE,
    &EventConditions            EVENT-CONDITION,
    &EventActions               EVENT-ACTION,
    &EventEnrollments           EVENT-ENROLLMENT,
    &EventConditionLists        EVENT-CONDITION-LIST,
    &Journals                   JOURNAL,
    &services                   ServiceSupportOptions,
    &parameters                 ParameterSupportOptions,
    &nest                       INTEGER,
    &Ulsms                      ULSM,
IF (csr cspi)
    &extendedServices           AdditionalSupportOptions,
ENDIF
IF (cspi)
    &extendedParameters         AdditionalCBBOptions
ENDIF
    }
```

Instances of this object class are created by the Initiate service (see clause 8.4.1).

8.1.3.1 &aaIdentifier

This field identifies the application association. Since the value of this field is never communicated, its form is a local matter.

8.1.3.2 &client

This field serves to identify the MMS client with whom the association has been established. It is defined in annex A.

8.1.3.3 &abstractSyntax

This field identifies the abstract syntax in use on this association.

8.1.3.4 &authenticationValue

This field is the value of the Authentication Value as presented by the MMS client at application association establishment. The form of this field is governed by the facilities of the network supporting the MMS association (see annex A).

8.1.3.5 &Transactions

This field is the set of Transaction objects associated with this application association. The maximum number of objects in this set is established by the Initiate service.

8.1.3.6 &NamedVariables

This field identifies the Named Variable objects whose name scope is AA-specific. This field is present only if the **vnam** parameter CBB has been negotiated. Named Variables are described in clause 14.

8.1.3.7 &NamedVariableLists

This field identifies the Named Variable List objects whose name scope is application-specific. This field is present only if the **vnam** and **vlis** parameter CBBs have been negotiated. Named Variables Lists are described in clause 14.

8.1.3.8 &NamedTypes

This field identifies the Named Type objects whose name scope is AA-specific. This field is present only if the **vnam** parameter CBB has been negotiated. Named Types are described in clause 14.

8.1.3.9 &DataExchanges

This field identifies the Data Exchange objects whose name scope is AA-specific. Data Exchanges are described in clause 15.

8.1.3.10 &EventConditions

This field identifies the Event Condition objects whose name scope is AA-specific. Event Conditions are described in clause 19 and event processing is described in clause 18.

8.1.3.11 &EventActions

This field identifies the Event Action objects whose name scope is AA-specific. Event Actions are described in clause 20 and event processing is described in clause 18.

8.1.3.12 &EventEnrollments

This field identifies the Event Enrollment objects whose name scope is AA-specific. Event Enrollments are described in clause 21 and event processing is described in clause 18.

8.1.3.13 &EventConditionLists

This field identifies the Event Condition List objects whose name scope is AA-specific. Event Condition Lists are described in clause 22 and event processing is described in clause 18.

8.1.3.14 &Journals

This field identifies the Journal objects whose name scope is AA-specific. Journals are described in clause 23.

8.1.3.15 &services

This field identifies the MMS services supported on this association. Each bit identifier corresponds to a specific service, either confirmed or unconfirmed. The value of this field is established as part of the Initiate procedure (see 8.2).

```
ServiceSupportOptions ::= BIT STRING {
    status                  (0),
    getNameList             (1),
    identify                (2),
    rename                  (3),
```

```
        read                                (4),
        write                               (5),
        getVariableAccessAttributes         (6),
        defineNamedVariable                 (7),
 -- bit 8 is reserved for use of a service defined in annex E
        defineScatteredAccess               (8),
 -- bit 9 is reserved for use of a service defined in annex E
        getScatteredAccessAttributes        (9),
        deleteVariableAccess                (10),
        defineNamedVariableList             (11),
        getNamedVariableListAttributes      (12),
        deleteNamedVariableList             (13),
        defineNamedType                     (14),
        getNamedTypeAttributes              (15),
        deleteNamedType                     (16),
        input                               (17),
        output                              (18),
        takeControl                         (19),
        relinquishControl                   (20),
        defineSemaphore                     (21),
        deleteSemaphore                     (22),
        reportSemaphoreStatus               (23),
        reportPoolSemaphoreStatus           (24),
        reportSemaphoreEntryStatus          (25),
        initiateDownloadSequence            (26),
        downloadSegment                     (27),
        terminateDownloadSequence           (28),
        initiateUploadSequence              (29),
        uploadSegment                       (30),
        terminateUploadSequence             (31),
        requestDomainDownload               (32),
        requestDomainUpload                 (33),
        loadDomainContent                   (34),
        storeDomainContent                  (35),
        deleteDomain                        (36),
        getDomainAttributes                 (37),
        createProgramInvocation             (38),
        deleteProgramInvocation             (39),
        start                               (40),
        stop                                (41),
        resume                              (42),
        reset                               (43),
        kill                                (44),
        getProgramInvocationAttributes      (45),
        obtainFile                          (46),
        defineEventCondition                (47),
        deleteEventCondition                (48),
        getEventConditionAttributes         (49),
        reportEventConditionStatus          (50),
        alterEventConditionMonitoring       (51),
        triggerEvent                        (52),
        defineEventAction                   (53),
        deleteEventAction                   (54),
        getEventActionAttributes            (55),
        reportEventActionStatus             (56),
        defineEventEnrollment               (57),
        deleteEventEnrollment               (58),
        alterEventEnrollment                (59),
        reportEventEnrollmentStatus         (60),
        getEventEnrollmentAttributes        (61),
        acknowledgeEventNotification        (62),
        getAlarmSummary                     (63),
        getAlarmEnrollmentSummary           (64),
```

```
        readJournal                     (65),
        writeJournal                    (66),
        initializeJournal               (67),
        reportJournalStatus             (68),
        createJournal                   (69),
        deleteJournal                   (70),
        getCapabilityList               (71),
    -- bit 72 is reserved for use of a service defined in annex D
        fileOpen                        (72),
    -- bit 73 is reserved for use of a service defined in annex D
        fileRead                        (73),
    -- bit 74 is reserved for use of a service defined in annex D
        fileClose                       (74),
    -- bit 75 is reserved for use of a service defined in annex D
        fileRename                      (75),
    -- bit 76 is reserved for use of a service defined in annex D
        fileDelete                      (76),
    -- bit 77 is reserved for use of a service defined in annex D
        fileDirectory                   (77),
        unsolicitedStatus               (78),
        informationReport               (79),
        eventNotification               (80),
        attachToEventCondition          (81),
        attachToSemaphore               (82),
        conclude                        (83),
        cancel                          (84),
        getDataExchangeAttributes       (85),
            -- Shall not appear in minor version one
        exchangeData                    (86),
            -- Shall not appear in minor version one
        defineAccessControlList         (87),
            -- Shall not appear in minor version one or two
        getAccessControlListAttributes  (88),
            -- Shall not appear in minor version one or two
        reportAccessControlledObjects   (89),
            -- Shall not appear in minor version one or two
        deleteAccessControlList         (90),
            -- Shall not appear in minor version one or two
        alterAccessControl              (91),
            -- Shall not appear in minor version one or two
        reconfigureProgramInvocation    (92) }
```

8.1.3.16 &parameters

This field identifies the MMS parameter CBBs that have been negotiated in the Initiate procedure (see 8.2).

```
ParameterSupportOptions ::= BIT STRING {
        str1                    (0),
        str2                    (1),
        vnam                    (2),
        valt                    (3),
        vadr                    (4),
    -- bit 5 is reserved
        vsca                    (5),
        tpy                     (6),
        vlis                    (7),
    -- bit 8 is reserved
    -- bit 9 is reserved
        cei                     (10),
        aco                     (11),
        sem                     (12),
        csr                     (13),
        csnc                    (14),
```

csplc (15),
cspi (16) }

8.1.3.16.1 str1

The **str1** parameter conformance building block shall establish the validity of the ARRAY value for the Kind Of Type parameter of the Type Description parameter, the INDEX and INDEX-RANGE value for the Access Selection parameters of the Alternate Access parameter (if **valt** is supported) and the ARRAY value for the Kind Of Data parameter of the Data parameter.

If **str1** is supported, these values, and the parameters that they select, are valid and the &nest field shall specify a value greater than zero. Otherwise, these values, and the parameters that they select, are invalid.

8.1.3.16.2 str2

The **str2** parameter conformance building block shall establish the validity of the STRUCTURE value for the Kind Of Type parameter of the Type Description parameter, the COMPONENT value for the Access Selection parameters of the Alternate Access parameter (if **valt** is supported) and the STRUCTURE value for the Kind Of Data parameter of the Data parameter.

If **str2** is supported, these values, and the parameters that they select, are valid and the &nest field shall specify a value greater than zero. Otherwise, these values, and the parameters that they select, are invalid.

8.1.3.16.3 vnam

The **vnam** parameter conformance building block shall establish the validity of the NAMED value for the Kind Of Variable parameter of the Variable Specification parameter and the NAMED value for the Kind Of Variable parameter of the GetVariableAccessAttributes service's Argument parameter.

If **vnam** is supported, these values and the parameters that they select are valid. Otherwise, these values and the parameters that they select are invalid.

8.1.3.16.4 valt

The valt parameter conformance building block shall establish the validity of the Alternate Access and Component Name parameters, wherever they occur in a service table.

If **valt** is supported, these parameters are valid. Otherwise, these parameters are invalid.

8.1.3.16.5 vadr

The **vadr** parameter conformance building block shall establish the validity of the Address parameter, wherever it occurs in a service table, and the validity of the UNNAMED and SINGLE values for the Kind Of Variable parameter of the Variable Specification parameter, the UNNAMED value for the Kind Of Variable parameter of the GetVariableAccessAttributes service's Argument parameter, and the value true for the Packed parameter of the Array and Structure parameters of the Type Description parameter.

If **vadr** is supported, the Address parameter and the specified values and the parameters that they select are valid. Otherwise, the Address parameter and the specified values and the parameters that they select are invalid.

8.1.3.16.6 vsca

The **vsca** CBB identifies facilities and services present in the first edition but not included in the object model in clause 7. These facilities and services are described in annex E.

8.1.3.16.7 tpy

The **tpy** parameter conformance building block shall indicate the functional capability to communicate with a third party. The third party may be an MMS client, an MMS server, or any other system. This third party system may be used by the MMS server to obtain resources required to execute an MMS service request. The choice of the method of communication with the third party is a local matter.

If **tpy** is supported, this value and the parameters that it selects are valid. Otherwise, this value and the parameters that it selects are invalid.

8.1.3.16.8 vlis

The **vlis** parameter conformance building block shall establish the validity of the Variable List Name value for the Kind of Access parameter of the Variable Access Specification parameter.

If **vlis** is supported, this value and the parameters that it selects are valid. Otherwise, this value and the parameters that it selects are invalid.

8.1.3.16.9 cei

The **cei** parameter conformance building block shall establish the validity of the Evaluation Interval of the AlterEventConditionMonitoring service's Argument parameter.

If **cei** is supported, this parameter is valid. Otherwise, it is invalid.

8.1.3.16.10 aco

The **aco** parameter conformance building block shall establish the validity of the Access Control parameter, wherever it occurs in a service table.

If **aco** is supported, this parameter is valid. Otherwise, it is invalid.

8.1.3.16.11 sem

The **sem** parameter conformance building block shall establish the validity of the Meaning parameter in the GetVariableAccessAttributes and GetNamedTypeAttributes services' Result(+) parameter.

If **sem** is supported, this parameter is valid. Otherwise, it is invalid.

8.1.3.16.12 csr

The **csr** parameter conformance building block serves to identify parameters of extensions to MMS services first introduced in ISO/IEC 9506-3.

If **csr** is supported, these parameters and the parameters they select are valid. Otherwise, these parameters and the parameters they select are invalid.

8.1.3.16.13 csnc

The **csnc** parameter conformance building block serves to identify parameters of extensions to MMS services first introduced in ISO/IEC 9506-4.

If **csnc** is supported, these parameters and the parameters they select are valid. Otherwise, these parameters and the parameters they select are invalid.

8.1.3.16.14 csplc

The **csplc** parameter conformance building block serves to identify parameters of extensions to MMS services first introduced in ISO/IEC 9506-5.

If **csplc** is supported, these parameters and the parameters they select are valid. Otherwise, these parameters and the parameters they select are invalid.

8.1.3.16.15 cspi

The **cspi** parameter conformance building block serves to identify parameters of extensions to MMS services first introduced in ISO/IEC 9506-6.

If **cspi** is supported, these parameters and the parameters they select are valid. Otherwise, these parameters and the parameters they select are invalid.

8.1.3.17 &nest

This field shall specify the maximum number of non-leaf nodes of a type tree between the root node and the most deeply nested leaf node. This value shall be zero if neither **str1** nor **str2** are supported. Otherwise, it shall be greater than zero and shall apply equally to **str1** and **str2**.

8.1.3.18 &Ulsms

This field identifies zero or more Upload State Machines objects. Upload State Machines are described in clause 11.

8.1.3.19 &extendedServices

This field identifies the extended MMS services supported on this association. This field is present only if one or more of the parameter CBB's **csr**, **csnc**, **csplc** or **cspi** have been negotiated. The value of this field is established by the Initiate procedure (see 8.2).

```
AdditionalSupportOptions ::= BIT STRING {
    vMDStop                              (0),
    vMDReset                             (1),
    select                               (2),
    alterProgramInvocationAttributes     (3),
    initiateUnitControlLoad              (4),
    unitControlLoadSegment               (5),
    unitControlUpload                    (6),
    startUnitControl                     (7),
    stopUnitControl                      (8),
    createUnitControl                    (9),
    addToUnitControl                     (10),
    removeFromUnitControl                (11),
    getUnitControlAttributes             (12),
    loadUnitControlFromFile              (13),
    storeUnitControlToFile               (14),
    deleteUnitControl                    (15),
    defineEventConditionList             (16),
    deleteEventConditionList             (17),
    addEventConditionListReference       (18),
    removeEventConditionListReference    (19),
    getEventConditionListAttributes      (20),
    reportEventConditionListStatus       (21),
    alterEventConditionListMonitoring    (22) }
```


8.1.3.20 &extendedParameters

This field identifies the extended MMS parameter CBBs that have been negotiated in the Initiate procedure (see 8.2). This field is present only if the **cspi** parameter CBB has been negotiated.

```
AdditionalCBBOptions ::= BIT STRING {
    des         (0),
    dei         (1),
    recl        (2) }
```

8.1.3.20.1 des

The **des** parameter conformance building block shall establish the validity of the string form of the Display Enhancement parameter, wherever it occurs in the service tables.

If **des** is supported, this parameter is valid. Otherwise, it is invalid.

8.1.3.20.2 dei

The **dei** parameter conformance building block shall establish the validity of the integer form of the Display Enhancement parameter, wherever it occurs in the service tables.

If **dei** is supported, this parameter is valid. Otherwise, it is invalid.

8.1.3.20.3 recl

The **recl** parameter conformance building block shall establish the validity of the List of Event Condition List names parameter, wherever it occurs in the service tables.

If **recl** is supported, this parameter is valid. Otherwise, it is invalid.

8.2 Initiate service

The Initiate service shall be used to establish the MMS environment and to allow the communicating MMS-users to exchange information regarding their capabilities and requirements. The Initiate service shall complete successfully before any other services may be carried out between a pair of MMS-users in the MMS environment. Annex A describes how the MMS Initiate service makes use of the underlying communication services.

8.2.1 Structure

The structure of the component service primitives is shown in Table 10.

Table 10 - Initiate service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M			
Local Detail Calling	U	U(=)			
Proposed Max Serv Outstanding Calling	M	M			
Proposed Max Serv Outstanding Called	M	M			
Proposed Data Structure Nesting Level	U	U			
Proposed Version Number	M	M			
Proposed Parameter CBB	M	M			
Proposed Additional Parameter CBB	C	C			
Services Supported Calling	M	M			
Extended Services Supported Calling	C	C			
Result(+)			S	S(=)	
Local Detail Called			U	U(=)	
Negotiated Max Serv Outstanding Calling			M	M(=)	
Negotiated Max Serv Outstanding Called			M	M(=)	
Negotiated Data Structure Nesting Level			U	U(=)	
Negotiated Version Number			M	M(=)	
Negotiated Parameter CBB			M	M(=)	
Negotiated Additional Parameter CBB			C	C(=)	
Services Supported Called			M	M(=)	
Extended Services Supported Called			C	C(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

8.2.1.1 Argument

This parameter shall convey the service specific parameters of the Initiate service request.

For those parameters in the request primitive, the MMS-provider in the calling system may reduce the values supplied by the calling MMS-user. For those parameters in the indication primitive, the MMS-provider in the called system may reduce the values from those in the InitiateRequestPDU, except for the Services Supported Calling and Extended Services Supported Calling parameters. Reductions of the values must follow the rules for reduction as specified in the parameter descriptions. No other modifications are permitted to these values.

NOTE The purpose of allowing the service providers to reduce values as specified by the user is to provide a mechanism that may be used to ensure that the values specified by the user do not exceed the capabilities of the service provider. Implementation of this feature is a local matter.

8.2.1.1.1 Local Detail Calling

When present, this parameter, of type integer, shall represent information about the calling MMS-user's implementation. The content of this field is a local matter and not a subject for further standardization.

8.2.1.1.2 Proposed Max Serv Outstanding Calling

This parameter, of type integer, shall identify the proposed maximum number of Transaction object instances in the &Transaction field of the Application Association object created at the calling MMS-user.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive. The value in the request or indication primitives shall not be less than zero.

8.2.1.1.3 Proposed Max Serv Outstanding Called

This parameter, of type integer, shall identify the proposed maximum number of Transaction object instances that may be created at the called MMS-user in the &Transaction field of the Application Association object.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive. The value in the request or indication primitives shall not be less than zero.

8.2.1.1.4 Proposed Data Structure Nesting Level

This parameter, of type integer, shall indicate the proposed maximum number of levels of nesting supported by both of the MMS-users that may occur within any data element used in the association. Absence of this parameter shall indicate an unlimited number of nesting levels.

The request and indication primitives shall specify the maximum nesting level of Type Specification (explicit or derived) that the calling MMS-user desires to use in the negotiated MMS environment. A value of zero shall indicate that only simple types are allowed.

The value of this parameter may be reduced by the MMS-provider. The value in the indication primitive shall be less than or equal to the value in the request primitive.

8.2.1.1.5 Proposed Version Number

This parameter, of type integer, shall contain a number that represents a minor version number of ISO 9506-1 and ISO 9506-2. The minor version number of ISO 9506-1 and ISO 9506-2 shall be that specified in ISO 9506-2, clause 24. This parameter, the Proposed Version Number, is the proposed minor version number that will be used for this instance of communication. Proposal of a number greater than one indicates support on this application association for all minor versions between one and the number proposed, inclusive.

NOTE Major versions of ISO 9506-1 and ISO 9506-2 are reflected through the definition and registration of distinct abstract syntaxes. Minor versions are reflected in the minor version number parameter. Minor versions of ISO 9506-1 and ISO 9506-2 at the same major version level are compatible with major versions of ISO 9506-1 and ISO 9506-2 with smaller minor version numbers.

The value of this parameter may be reduced by the MMS-provider if it cannot support the requested value. The value in the indication primitive shall be less than or equal to the value in the request primitive, but not less than one.

8.2.1.1.6 Proposed Parameter CBB

This parameter, of type bitstring, shall specify the set of parameter conformance building blocks (CBB) that are proposed to be supported on this application association.

The value of this parameter in the request primitive shall specify the set of Parameter CBBs supported by the Calling MMS-user.

The value of this parameter in the indication primitive shall specify the intersection of the set of Parameter CBBs supported by the Calling MMS-user and the set of Parameter CBBs supported by the MMS-provider.

The interpretation of a parameter CBB and assignment to an individual bit of a CBB bitstring type is specified in 8.1.3.16. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

NOTE The set of Parameter CBBs used in this parameter only contains those that can be represented by a binary value. The value of the NEST Parameter CBB is conveyed in the Proposed Data Structure Nesting Level parameter.

8.2.1.1.7 Proposed Additional Parameter CBB

This parameter, of type bitstring, shall specify the set of additional parameter conformance building blocks (CBB) that are proposed to be supported on this application association. This parameter shall be present only if the **cspi** parameter CBB has been offered in the Proposed Parameter CBB parameter.

The value of this parameter in the request primitive shall specify the set of Additional Parameter CBBs supported by the Calling MMS-user.

The value of this parameter in the indication primitive shall specify the intersection of the set of Additional Parameter CBBs supported by the Calling MMS-user and the set of Additional Parameter CBBs supported by the MMS-provider.

The interpretation of an additional parameter CBB and assignment to an individual bit of a CBB bitstring type is specified in 8.1.3.20. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

8.2.1.1.8 Services Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS-user of a set of services for use on this application association.

The value of the parameter in the indication primitive shall specify the intersection of the set of services supported by the Calling MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in 8.1.3.15. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the Modifier and to execute properly the service procedure defined for the Modifier.

If a confirmed service, an unconfirmed service, or Modifier is supported, a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, a Reject PDU shall be issued on receipt of that service request or modified service request with a reject code of "UNRECOGNIZED SERVICE".

8.2.1.1.9 Extended Services Supported Calling

This parameter, of type bitstring, shall specify support by the Calling MMS-user of a set of extended services for use on this application association. This parameter shall not be present unless either the **csr** parameter CBB or the **cspi** parameter CBB (or both) have been offered in the Proposed Parameter CBB parameter.

The value of the parameter in the indication primitive shall specify the intersection of the set of services supported by the Calling MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in 8.1.3.19. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive a request indication and execute the service procedure defined for the responder role.

If a confirmed service is supported, a Reject PDU shall not be issued on receipt of that service, except in the case of a protocol error. If a confirmed service is not supported, a Reject PDU shall be issued on receipt of that service request with a reject code of "UNRECOGNIZED SERVICE".

8.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

8.2.1.2.1 Local Detail Called

When present, this parameter, of type integer, shall represent information about the Called MMS-User's implementation. The content of this field is a local matter and not a subject for further standardization.

8.2.1.2.2 Negotiated Max Serv Outstanding Calling

This parameter, of type integer, shall identify the maximum number of Transaction object instances that may be created at the calling MMS-user in the &Transaction field of the Application Association object.

The negotiated maximum number of services outstanding in the response primitive shall be less than or equal to the Proposed Max Serv Outstanding Calling parameter in the indication primitive, but shall not be less than zero.

8.2.1.2.3 Negotiated Max Serv Outstanding Called

This parameter, of type integer, shall identify the maximum number of Transaction object instances that may be created at the called MMS-user in the &Transaction field of the Application Association object.

The negotiated maximum number of services outstanding in the response primitive shall be less than or equal to the Proposed Max Serv Outstanding Called parameter in the indication primitive, but shall not be less than zero.

8.2.1.2.4 Negotiated Data Structure Nesting Level

This parameter, of type integer, shall indicate the maximum number of levels of nesting supported by both of the MMS-users that may occur within any data element used in the association. Absence of this parameter shall indicate an unlimited nesting level.

This parameter shall be equal to or less than the Proposed Data Structure Nesting level parameter in the indication primitive, but shall not be less than zero.

The response and confirmation primitives shall specify the maximum nesting level of Type Specification (explicit or derived) that shall be used in the negotiated MMS environment. A value of zero shall indicate that only simple types are allowed.

8.2.1.2.5 Negotiated Version Number

This parameter, of type integer, shall contain a number that represents a minor version number of ISO 9506-1 and ISO 9506-2. The minor version number of ISO 9506-1 and ISO 9506-2 shall be that specified in ISO 9506-2, clause 24. This parameter, the Negotiated Version Number, shall be the minor version number that will be used for this instance of communication. This number shall be less than or equal to the Proposed Version Number parameter in the request primitive. It shall not be less than one.

NOTE Major versions of ISO 9506 are reflected through the definition and registration of distinct abstract syntaxes. Minor versions are reflected in the minor version number parameter. Minor versions of ISO 9506-1 and ISO 9506-2 at the same major version level are compatible with versions of ISO 9506-1 and ISO 9506-2 with smaller minor version numbers.

8.2.1.2.6 Negotiated Parameter CBB

This parameter, of type bitstring, shall specify the negotiated set of parameter conformance building blocks (CBB) that are to be supported on this application association.

The value of this parameter in the response primitive shall specify the intersection of the set of Parameter CBBs supported by the Called MMS-user and the set of Parameter CBBs specified by the Proposed Parameter CBB parameter in the indication primitive.

The interpretation of a parameter CBB and assignment to an individual bit of a CBB bitstring type is specified in 8.1.3.16. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

NOTE The set of Parameter CBBs used in this parameter only contains those that can be represented by a binary value. The value of the &next field is conveyed in the Negotiated Data Structure Nesting Level parameter.

8.2.1.2.7 Negotiated Additional Parameter CBB

This parameter, of type bitstring, shall specify the negotiated set of additional parameter conformance building blocks (CBB) that are to be supported on this application association. This parameter shall be present only if the **cspi** parameter CBB has been negotiated in the Negotiated Parameter CBB parameter.

The value of this parameter in the response primitive shall specify the intersection of the set of Additional Parameter CBBs supported by the Called MMS-user and the set of Additional Parameter CBBs specified by the Proposed Additional Parameter CBB parameter in the indication primitive.

The possible parameter conformance building blocks are specified in ISO 9506-2, clause 18. The assignment of a Parameter CBB to an individual bit of a CBB bitstring type is specified in 8.1.3.20. A value of one in the assigned bit shall indicate support for the corresponding CBB. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

8.2.1.2.8 Services Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS-user of a set of services for use on this application association.

The value of the parameter in the confirmation primitive shall specify the intersection of the set of services supported by the Called MMS-user and the set of services supported by the MMS-provider.

The assignment of a service to an individual bit of the bitstring type is specified in 8.1.3.15. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive an indication primitive and properly execute the service procedure defined for the responder role. Support of unconfirmed services shall be defined as the ability to accept an indication primitive and to pass the parameters to the service interface. Support of a modifier shall be defined as the ability to accept an indication primitive that contains the Modifier and to properly execute the service procedure defined for the Modifier.

If a confirmed service, an unconfirmed service, or Modifier is supported, a Reject PDU shall not be issued on receipt of that service or modified service, except in the case of a protocol error. If a confirmed service, an unconfirmed service, or modifier is not supported, a Reject PDU shall be issued on receipt of that service or modified service with a reject code of "UNRECOGNIZED SERVICE".

8.2.1.2.9 Extended Services Supported Called

This parameter, of type bitstring, shall specify support by the Called MMS-user of a set of additional services for use on this application association.

The value of the parameter in the confirmation primitive shall specify the intersection of the set of additional services supported by the Called MMS-user and the set of additional services supported by the MMS-provider.

The assignment of an additional service to an individual bit of the bitstring type is specified in 8.1.3.19. A value of one in the assigned bit shall indicate support for the corresponding service. A value of zero shall indicate non-support. All bits shall be encoded and any additional bits received shall be ignored.

Support for confirmed services shall be defined as the ability to receive an indication primitive and properly execute the service procedure defined for the responder role.

If a confirmed service is supported, a Reject PDU shall not be issued on receipt of that service, except in the case of a protocol error. If a confirmed service is not supported, a Reject PDU shall be issued on receipt of that service with a reject code of "UNRECOGNIZED SERVICE".

8.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

8.2.2 Service Procedure

The called MMS-user shall issue a response primitive indicating success in the Result parameter if that MMS-user is willing to accept communications in the MMS environment under the constraints identified in the indication primitive, or if alternate values can be proposed (according to the negotiation rules), with the requesting MMS-user. Otherwise, a response primitive indicating the Result(-) parameter shall be issued.

Successful execution of the Initiate service shall result in the establishment of the MMS environment. The MMS environment shall only be established through the use of the Initiate service. The Initiate service shall not be used within an established MMS environment. If an Initiate Request PDU is received on an association where an established MMS environment exists, a Reject PDU shall be issued with a Reject PDU Type of PDU-ERROR and a Reject Code of ILLEGAL-ACSE-MAPPING.

As a result of a successful execution of the Initiate service, an Application Association object shall be created and initialised as follows:

a) The &aaIdentifier field shall be assigned an unique integer.

b) The &client field shall be set to the Application Reference of the peer MMS-user.

c) The &abstractSyntax field shall be set to the abstract syntax used for communication.

d) The &authenticationValue field shall be set to the parameter of M-ASSOCIATE service, if present; otherwise it shall be empty.

e) The &Transactions field shall be empty.

f) The &services field shall be set to the value of the Services Supported Calling parameter of the Initiate.indication service primitive at the called MMS-user, and to the value of the Services Supported Called parameter of the Initiate.confirm(+) service primitive at the calling MMS-user.

g) The &parameters field shall be set to the value of the Negotiated Parameter CBB parameter of the Initiate.confirm service primitive at both the calling and the called MMS-users.

h) The &nest field shall be set to the value of the Negotiated Data Structure Nesting Level parameter of the Initiate.confirm service primitive at both the calling and the called MMS-users.

i) The extendedServices field shall be set to the value of the Extended Services Supported Calling parameter of the Initiate.indication service primitive at the called MMS-user, and to the value of the Extended Services Supported Called parameter of the Initiate.confirm(+) service primitive at the calling MMS-user.

j) The &extendedParameters field shall be set to the value of the Negotiated Additional Parameter CBB parameter of the Initiate.confirm service primitive at both the calling and the called MMS-users.

k) Other object set fields of the Application Association shall be initialised to empty or as reported in the Configuration and Initialisation (see ISO 9506-2, clause 25).

8.3 Conclude service

The Conclude service may be used to cause the orderly relinquishing of the MMS environment. An MMS-user requests the Conclude service to indicate its willingness to disestablish the Application Association. Annex A describes how the MMS Conclude service makes use of the underlying communication services.

8.3.1 Structure

The structure of the component service primitives is shown in Table 11.

Table 11 - Conclude service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Result(+)			S	S(=)	
Result(-) Error Type			S M	S(=) M(=)	

8.3.1.1 Argument

There are no service specific parameters for the Conclude service request.

8.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

8.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

8.3.2 Service Procedure

8.3.2.1 Requesting MMS-user

The requesting MMS-user shall issue a conclude.request service primitive, which begins the Conclude service. Once this primitive has been issued, the requesting MMS-user shall not issue any further request primitives until a conclude.confirm service primitive

is received from the MMS-provider (except the Abort request primitive, which may be issued at any time). The requesting MMS-user may continue to issue response primitives in order to complete service requests from the peer MMS-user.

The requesting MMS-user shall not issue a conclude.request service primitive to a peer MMS-user on an association if a Domain exists at the requesting MMS-user with a &state field value equal to **loading**, **complete**, **incomplete**, **d1**, **d2**, **d3**, or **d9** and a &aAssociation field indicating this association.

Upon receiving a conclude.confirm service primitive with a Result(+), the requesting MMS-user shall consider the MMS environment to be terminated, and no further request or response primitives shall be issued by the requesting MMS-user in the MMS environment on that association.

Upon receiving a conclude.confirm service primitive with a Result(-), the requesting MMS-user shall consider the MMS Environment to be unaffected, and may continue to issue request or response service primitives in the MMS environment.

Upon successful completion of the Conclude service, all AA-specific objects at the requesting MMS-user associated with the terminated MMS environment are released and any defined procedures associated with their deletion are performed unless other specifications for that object are explicitly stated in this part of ISO 9506.

8.3.2.2 Responding MMS-user

Upon receipt of the conclude.indication service primitive, the responding MMS-user shall issue a conclude.response service primitive indicating whether or not it accepts the termination of the association before any other service request primitives may be issued by that MMS-user (except the abort.request service primitive, which may be issued at any time). The responding MMS-user shall not accept conclusion on an association if it is awaiting any response from the peer MMS-user for a confirmed service or for the Conclude Service.

A responding MMS-user shall not accept a Conclude attempt on an association if any of the following conditions exist:

a) The responding MMS-user has received an indication on the association for a confirmed service request from the peer MMS-user for which it has not yet issued a response;

b) An Upload State Machine exists on the responding MMS-user as a result of a request from the peer MMS-user on the association;

c) A Domain exists at the responding MMS-user with a &state field value equal to **loading**, **complete**, or **incomplete** and an &aAssociation field indicating this association.

d) The responding MMS-user has control of a semaphore on the requesting MMS-user on the association;

e) The requesting MMS-user has control of a semaphore on the responding MMS-user on the association.

Upon issuing a conclude.response service primitive with a Result(+) parameter, the MMS Environment shall be considered terminated. No further request or response primitives shall be issued by the responding MMS-user in that MMS environment on that association.

Upon issuing a conclude.response service primitive with a Result(-), the MMS environment shall not be affected and the responding MMS-user may continue to issue request and/or response service primitives.

NOTE 1 The effect of a failed Conclude service confirmation is as if no Conclude request had ever been issued.

Upon successful completion of the Conclude service, all AA-specific objects at the responding MMS-user associated with the terminated MMS environment are released and any defined procedures associated with their deletion are performed unless other specifications for that object are explicitly stated in this part of ISO 9506.

NOTE 2 It is possible that both of the peers will initiate a Conclude attempt at or near the same time resulting in failure of both attempts. If both peers retry the Conclude, on result of failure, this situation could potentially (although very unlikely) continue indefinitely.

Through use of application design, this situation can be avoided. One possible method is to designate the calling MMS-user as the peer that will initiate the Conclude attempt should this situation arise.

8.4 Abort service

The Abort service shall be used to relinquish the MMS environment abruptly and without negotiation. The MMS-user shall issue the abort.request service primitive to indicate that it wishes immediately, and without negotiation, to discontinue communications in the MMS Environment on the application association. The effect of the Abort service may be to destroy previously issued requests and/or responses issued by either of the MMS-users. Annex A describes how the MMS Abort service makes use of the underlying communication services.

NOTE The abort.indication service primitive may also be generated by the MMS-provider.

8.4.1 Structure

The structure of the component service primitives is shown in Table 12.

Table 12 - Abort service

Parameter Name	Req	Ind	CBB
Argument	M	M	
Locally Generated		M	

8.4.1.1 Argument

This parameter shall convey the service specific parameters of the Abort service.

8.4.1.1.1 Locally Generated

This parameter, of type boolean, shall indicate whether the abort request was generated by the system in which the MMS-user receiving the indication is located (indicated by the value true), or whether the abort request was received by that system (indicated by the value false). This parameter shall be provided by the MMS-provider.

8.4.2 Service Procedure

In the case of an Abort initiated by the MMS-user, the peer MMS-user shall be notified of the user requested abort by receipt of an abort.indication service primitive, and the MMS Environment shall be terminated.

In the case of an Abort initiated by the MMS-provider, both MMS-users shall be notified of the provider abort by receipt of abort.indication primitives (if this is possible), and the MMS Environment shall be terminated.

Upon completion of the Abort service, all AA-specific objects associated with the terminated MMS environment both MMS-users are released and any defined procedures associated with their deletion are performed unless other specifications for that object are explicitly stated in this part of ISO 9506.

8.5 Cancel service

The Cancel service is used by an MMS client to cancel a request that has previously been issued, but has not yet completed. Only confirmed services may be cancelled (see clause 5).

8.5.1 Structure

The structure of the component service primitives is shown in Table 13.

Table 13 - Cancel service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Original Invoke ID	M	M(=)			
Result(+)			S	S(=)	
Original Invode ID			M	M(=)	
Result(-)			S	S(=)	
Original Invoke ID			M	M(=)	
Error Type			M	M(=)	

8.5.1.1 Argument

This parameter shall convey the service specific parameters for the Cancel service request.

8.5.1.1.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service whose cancellation is desired. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

8.5.1.2.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service that was cancelled. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

8.5.1.3.1 Original Invoke ID

This parameter, of type integer, shall specify the invoke ID of the service whose cancellation was desired. (An invoke ID is provided for every confirmed service request primitive, see clause 5.)

8.5.2 Service Procedure

8.5.2.1 Preconditions

The following conditions shall result in a Response(-) to the Cancel request.

a) the service request identified by the Cancel request has not been received;

b) the service response for this Cancel request has already been issued;

c) the &cancelable field of the Transaction object associated with this service request has been set to false;

d) the service request can be cancelled non-destructively, i.e., the service can return to a state that is logically as if the service indication targeted for cancellation had not been received. Destructive cancellation is permitted for the Start, Stop, Resume, and Reset Services provided it follows the service procedures defined for these services.

For these cases, the responding MMS-user shall issue a Cancel response service primitive with the Result(-) parameter with the appropriate Cancel error codes for the Error Type parameter. The responding MMS-user shall continue with the original service and eventually return a response, either positive or negative, as appropriate.

8.5.2.2 Action

The service invocation identified shall be cancelled by the responding MMS-user, and a Cancel response primitive with the Result(+) parameter shall be issued. The responding MMS-user shall also issue a response primitive with a Result(-) parameter for the cancelled service indicating the SERVICE-PREEMPT Error Class with an Error Code of CANCEL. The state of the responder shall then appear logically as if the service targeted for cancellation had not been requested, unless otherwise stated in the service procedure for the cancelled service. Additional limitations on requesting the cancel service are specified in ISO 9506-2, clause 6.

8.6 Reject service

The Reject service is a provider-initiated service that is used to inform the MMS-users of the occurrence of a protocol error. A definition of a protocol error may be found in ISO 9506-2, clause 6.

8.6.1 Structure

The structure of the component service primitives is shown in Table 14.

Table 14 - Reject service

Parameter Name	Ind	CBB
Detected Here	M	
Original Invoke ID	C	
Reject PDU Type	M	
Reject Code	M	

8.6.1.1 Detected Here

This parameter, of type boolean, shall indicate whether the protocol error that results in the Reject service was detected at the local end or remote end of the application association. If true, the protocol error was detected at the local end; if false, the error was detected at the remote end.

NOTE Handling of local error conditions between an MMS-user and MMS-provider is a local matter.

8.6.1.2 Original Invoke ID

This parameter, of type integer, shall indicate the original invoke ID of the PDU that was found to be in error. Its presence is conditional on whether the invoke ID could be determined. If there is no invoke ID specified in the PDU being rejected, this parameter shall not be present.

8.6.1.3 Reject PDU Type

This parameter, of type integer, shall indicate the type of the PDU that caused the protocol error. The value PDU-ERROR shall be used when the PDU being rejected is not a syntactically valid MMS PDU. The possible values are as follows:

CONFIRMED-REQUESTPDU
CONFIRMED-RESPONSEPDU
CONFIRMED-ERRORPDU
UNCONFIRMEDPDU
PDU-ERROR
CANCEL-REQUESTPDU
CANCEL-RESPONSEPDU
CANCEL-ERRORPDU
CONCLUDE-REQUESTPDU
CONCLUDE-RESPONSEPDU
CONCLUDE-ERRORPDU

8.6.1.4 Reject Code

This parameter, of type integer, shall indicate additional information regarding the reason for the Reject within a given Reject PDU Type parameter value. The possible values for this parameter are given below under each possible Reject PDU Type parameter value.

8.6.1.4.1 Codes for CONFIRMED-REQUESTPDU

8.6.1.4.1.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.1.2 UNRECOGNIZED-SERVICE

This code shall be used when the service requested is not supported or is not recognized.

8.6.1.4.1.3 UNRECOGNIZED-MODIFIER

This code shall be used when a modifier requested is not supported or is not recognized.

8.6.1.4.1.4 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of this International Standard

8.6.1.4.1.5 INVALID-ARGUMENT

This code shall be used when a service argument does not meet the requirements of this part of ISO 9506.

8.6.1.4.1.6 INVALID-MODIFIER

This code shall be used when a service modifier does not meet the requirements of this part of ISO 9506.

8.6.1.4.1.7 MAX-SERV-OUTSTANDING-EXCEEDED

This code shall be used when the negotiated maximum number of confirmed services that may be outstanding is exceeded by a confirmed service request that is received.

8.6.1.4.1.8 MAX-RECURSION-EXCEEDED

This code shall be used when the PDU received exceeds the negotiated maximum data structure nesting level.

8.6.1.4.1.9 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by this part of ISO 9506 .

8.6.1.4.2 Codes for CONFIRMED-RESPONSEPDU

8.6.1.4.2.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.2.2 UNRECOGNIZED-SERVICE

This code shall be used when the service specified is not supported, is not recognized, or is not the same service that was requested with the invoke ID specified in the PDU.

8.6.1.4.2.3 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of this International Standard, or no confirmed service has been requested for the specified invoke ID.

8.6.1.4.2.4 INVALID-RESULT

This code shall be used when a service result does not meet the requirements of this part of ISO 9506.

8.6.1.4.2.5 MAX-RECURSION-EXCEEDED

This code shall be used when the PDU received exceeds the negotiated maximum data structure nesting level.

8.6.1.4.2.6 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by this part of ISO 9506.

8.6.1.4.3 Codes for CONFIRMED-ERRORPDU

8.6.1.4.3.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.3.2 UNRECOGNIZED-SERVICE

This code shall be used when the service specified is not supported, is not recognized, or is not the same service that was requested with the invoke ID specified in the PDU.

8.6.1.4.3.3 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of this International Standard, or no confirmed service has been requested for the specified invoke ID.

8.6.1.4.3.4 INVALID-SERVICEERROR

This code shall be used when a service error does not meet the requirements of this part of ISO 9506.

8.6.1.4.3.5 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by this part of ISO 9506.

8.6.1.4.4 Codes for UNCONFIRMEDPDU

8.6.1.4.4.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.4.2 UNRECOGNIZED-SERVICE

This code shall be used when the service specified is not supported or is not recognized.

8.6.1.4.4.3 INVALID-ARGUMENT

This code shall be used when a service argument does not meet the requirements of this part of ISO 9506.

8.6.1.4.4.4 MAX-RECURSION-EXCEEDED

This code shall be used when the PDU received exceeds the negotiated maximum data structure nesting level.

8.6.1.4.4.5 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by this part of ISO 9506.

8.6.1.4.5 Codes for PDU-ERROR

8.6.1.4.5.1 UNKNOWN-PDU-TYPE

This code shall be used when the PDU type received is not recognized or is not supported.

8.6.1.4.5.2 INVALID-PDU

This code shall be used when the PDU received is syntactically incorrect, and further diagnostics cannot be provided due to the severity of the error.

8.6.1.4.5.3 ILLEGAL-ACSE-MAPPING

This code shall be used when the PDU type received is not properly mapped to an ACSE service primitive.

8.6.1.4.6 Codes for CANCEL-REQUESTPDU

8.6.1.4.6.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.6.2 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of this International Standard.

8.6.1.4.7 Codes for CANCEL-RESPONSEPDU

8.6.1.4.7.1 OTHER

This code shall be used for errors other than those identified in this part of this part of ISO 9506 for this Reject PDU type.

8.6.1.4.7.2 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of this International Standard, or no Cancel service has been requested for the specified invoke ID.

8.6.1.4.8 Codes for CANCEL-ERRORPDU

8.6.1.4.8.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.8.2 INVALID-INVOKEID

This code shall be used when an invoke ID does not meet the requirements of this International Standard, or no Cancel service has been requested for the specified invoke ID.

8.6.1.4.8.3 INVALID-SERVICEERROR

This code shall be used when a service error does not meet the requirements of this part of ISO 9506.

8.6.1.4.8.4 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by this part of ISO 9506.

8.6.1.4.9 Codes for CONCLUDE-REQUESTPDU

8.6.1.4.9.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.9.2 INVALID-ARGUMENT

This code shall be used when the argument for the request does not meet the requirements of this part of ISO 9506.

8.6.1.4.10 Codes for CONCLUDE-RESPONSEPDU

8.6.1.4.10.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.10.2 INVALID-RESULT

This code shall be used when the service result does not meet the requirements of this part of ISO 9506.

8.6.1.4.11 Codes for CONCLUDE-ERRORPDU

8.6.1.4.11.1 OTHER

This code shall be used for errors other than those identified in this part of ISO 9506 for this Reject PDU type.

8.6.1.4.11.2 INVALID-SERVICEERROR

This code shall be used when a service error does not meet the requirements of this part of ISO 9506.

8.6.1.4.11.3 VALUE-OUT-OF-RANGE

This code shall be used when the PDU received contains one or more parameters whose values exceed the range allowed for those parameters by this part of ISO 9506.

8.6.2 Service Procedure

If an MMS-provider receives a RejectPDU it shall issue a reject.indication service primitive to the MMS-user. The MMS-user may, as a local matter, use the Abort service to terminate the MMS environment abruptly.

An MMS provider shall be capable of issuing a RejectPDU if it receives a PDU that constitutes a protocol error.

9 Conditioned service response

9.1 Introduction and Models

This clause provides a model for the following object:

ACCESS-CONTROL-LIST

This clause specifies the following services:

DefineAccessControlList
GetAccessControlListAttributes
ReportAccessControlledObjects
DeleteAccessControlList
ChangeAccessControl

This clause provides facilities in MMS that allow the specification of conditions under which certain MMS services are required to fail. Such facilities may be required, for example, to allow use of some MMS services by one client to the exclusion of other clients. This clause is consistent with the general MMS specification that does not require any MMS service request to succeed, but may require a service request to fail. Success is always dependent on conditions beyond the scope of MMS; failure may be required by the MMS service procedures.

The facilities of the Access Control List have been designed to permit interoperation with implementations conforming to earlier versions of this standard. The external manifestations of the Access Control List objects is (1) the reporting of the name of the Access Control List object referenced by a given object in its GetAccessControlListAttributes service, and (2) the use of the additional services specified in this clause. The first manifestation is governed by the negotiation of the **aco** parameter CBB; the second is governed by the corresponding service CBBs.

9.1.1 Access Control List Object Model

This clause introduces the model of the Access Control List object.

```
ACCESS-CONTROL-LIST ::= CLASS {
    &name                       Identifier, -- shall be unique within the VMD
    &accessControl              ACCESS-CONTROL-LIST,
    &readAccessCondition        [0] AccessCondition OPTIONAL,
    &storeAccessCondition       [1] AccessCondition OPTIONAL,
    &writeAccessCondition       [2] AccessCondition OPTIONAL,
    &loadAccessCondition        [3] AccessCondition OPTIONAL,
    &executeAccessCondition     [4] AccessCondition OPTIONAL,
    &deleteAccessCondition      [5] AccessCondition OPTIONAL,
    &editAccessCondition        [6] AccessCondition OPTIONAL,
    --
    -- The following fields are used to record lists of objects placed
    -- under the control of this ACCESS-CONTROL-LIST object.
    -- They will be referred to collectively as the Controlled Object Lists
    --
    &AccessControlLists         ACCESS-CONTROL-LIST,
    &Domains                    DOMAIN,
    &ProgramInvocations         PROGRAM-INVOCATION,
    &UnitControls               UNIT-CONTROL,
IF (vadr)
    &UnnamedVariables           UNNAMED-VARIABLE OPTIONAL,
ENDIF
IF (vnam)
    &NamedVariables             NAMED-VARIABLE,
IF (vlis)
    &NamedVariableLists         NAMED-VARIABLE-LIST,
ENDIF
    &NamedTypes                 NAMED-TYPE,
ENDIF
    &DataExchanges              DATA-EXCHANGE,
    &Semaphores                 SEMAPHORE,
    &OperatorStations           OPERATOR-STATION,
    &EventConditions            EVENT-CONDITION,
    &EventActions               EVENT-ACTION,
    &EventEnrollments           EVENT-ENROLLMENT,
    &Journals                   JOURNAL,
IF (cspi)
    &EventConditionLists        EVENT-CONDITION-LIST
ENDIF
    }
```

9.1.1.1 &name

The &name field uniquely identifies the Access Control List object within the VMD. The name shall be a VMD-specific Object Name formed according to the rules for MMS Object Names.

9.1.1.2 &accessControl

Each Access Control List object is itself subject to access control. This field identifies the Access Control List object that governs access to this object.

9.1.1.3 &readAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met for read access to the object. Access Condition is described in 9.1.2. The services affected are:

a) Read
b) Output

9.1.1.4 &storeAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met for store access to the object. Access Condition is described in 9.1.2. The services affected are:

a) ReadJournal
b) InitiateUploadSequence
c) StoreDomainContent
d) UnitControlUpload
e) StoreUnitControlToFile

9.1.1.5 &writeAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met for write access to the object. Access Condition is described in 9.1.2. The services affected are:

a) Write
b) Input
c) ExchangeData

9.1.1.6 &loadAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met for load access to the object. Access Condition is described in 9.1.2. The services affected are:

a) InitiateDownloadSequence*
b) LoadDomainContent*
c) CreateProgramInvocation*
d) DefineNamedVariable*
e) DefineNamedVariableList*
f) DefineNamedType*
g) DefineSemaphore
h) TakeControl
i) AttachToSemaphore
j) DefineEventCondition*
k) DefineEventAction*
l) DefineEventEnrollment*
m) TriggerEvent
n) AlterEventConditionMonitoring
o) AlterEventEnrollment
p) AcknowledgeEventNotification
q) CreateJournal*
r) InitializeJournal
s) WriteJournal
t) DefineAccessControlList*
u) AlterProgramInvocationAttributes
v) InitiateUnitControlLoad
w) CreateUnitControl*
x) DefineEventConditionList*
y) AlterEventConditionListMonitoring

NOTE * Since these services create these objects, the services can only be affected by the Access Control List referenced by the VMD.

9.1.1.7 &executeAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met for execute access to the object. Access Condition is described in 9.1.2. The services affected are:

a) Start
b) Stop
c) Resume
d) Reset
e) Kill
f) VMDStop
g) VMDReset
h) Select
i) StartUnitControl
j) StopUnitControl

9.1.1.8 &deleteAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met to delete the object. Access Condition is described in 9.1.2. The services affected are:

a) DeleteDomain
b) DeleteProgramInvocation
c) DeleteVariableAccess
d) DeleteNamedVariableList
e) DeleteNamedType
f) DeleteSemaphore
g) DeleteEventCondition
h) DeleteEventAction
i) DeleteEventEnrollment
j) DeleteJournal
k) DeleteAccessControlList
l) DeleteUnitControl
m) DeleteEventConditionList

9.1.1.9 &editAccessCondition

This field identifies the Access Condition instance that describes the access conditions to be met for edit access to the object, that is, to change the access control. Access Condition is described in 9.1.2. The services affected are:

a) ChangeAccessControl
b) Rename
c) AddToUnitControl
d) RemoveFromUnitControl
e) AddEventConditionListReference
f) RemoveEventConditionLIstReference

9.1.1.10 &AccessControlLists

This field identifies the set of Access Control List objects that have their access controlled by this object. Objects of this class are only affected by:

a) &deleteAccessCondition
b) &editAccessCondition

9.1.1.11 &Domains

This field identifies the set of Domain objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &storeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.1.12 &ProgramInvocations

This field identifies the set of Program Invocation objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &executeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.1.13 &UnitControls

This field identifies the set of Unit Control objects that have their access controlled by this object. This field is present only if the **cspi** parameter CBB has been negotiated. Objects of this class are only affected by:

a) &loadAccessCondition
b) &executeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.1.14 &UnnamedVariables

This field identifies the set of Unnamed Variable objects that have their access controlled by this object. This field is present only if the **vadr** parameter CBB has been negotiated. Objects of this class are only affected by:

a) &readAccessCondition
b) &writeAccessCondition

9.1.1.15 &NamedVariables

This field identifies the set of Named Variable objects that have their access controlled by this object. This field is present only if the **vnam** parameter CBB has been negotiated. Objects of this class are only affected by:

a) &readAccessCondition
b) &writeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.1.16 &NamedVariableLists

This field identifies the set of Named Variable List objects that have their access controlled by this object. This field is present only if the **vnam** and the **vlis** parameter CBB's have been negotiated. Objects of this class are only affected by:

a) &readAccessCondition
b) &writeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.1.17 &NamedTypes

This field identifies the set of Named Type objects that have their access controlled by this object. This field is present only if the **vnam** parameter CBB has been negotiated. Objects of this class are only affected by:

a) &deleteAccessCondition
b) &editAccessCondition

9.1.1.18 &DataExchanges

This field identifies the set of Data Exchange objects that have their access controlled by this object. Objects of this class are only affected by:

a) &writeAccessCondition
b) &editAccessCondition

9.1.1.19 &Semaphores

This field identifies the set of Semaphore objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &deleteAccessCondition
c) &editAccessCondition

9.1.1.20 &OperatorStations

This field identifies the set of Operator Station objects that have their access controlled by this object. Objects of this class are only affected by:

a) &readAccessCondition
b) &writeAccessCondition
c) &editAccessCondition

9.1.1.21 &EventConditions

This field identifies the set of Event Condition objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &deleteAccessCondition
c) &editAccessCondition

9.1.1.22 &EventActions

This field identifies the set of Event Action objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &deleteAccessCondition
c) &editAccessCondition

9.1.1.23 &EventEnrollments

This field identifies the set of Event Enrollment objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &deleteAccessCondition
c) &editAccessCondition

9.1.1.24 &Journals

This field identifies the set of Journal objects that have their access controlled by this object. Objects of this class are only affected by:

a) &loadAccessCondition
b) &storeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.1.25 &EventConditionLists

This field identifies the set of Event Condition List objects that have their access controlled by this object. This field is present only if the **cspi** parameter CBB has been negotiated. Objects of this class are only affected by:

a) &loadAccessCondition
b) &storeAccessCondition
c) &deleteAccessCondition
d) &editAccessCondition

9.1.2 Access Condition

This type specifies a condition that, if not satisfied, will require the MMS service to fail. This type is a choice of one of the following types:

a) never — This condition indicates that the service controlled by this Access Control List shall always fail.

b) semaphore — This condition indicates that if the named semaphore is not owned by the MMS service requester, the service controlled by this Access Control List shall fail.

c) user — This condition indicates that the service controlled by this Access Control List shall fail unless the Application Reference of the client matches the value of this field.

d) password This condition indicates that the service controlled by this Access Control List shall fail unless the client has provided the authentication values that match the value of this field.

e) joint The service controlled by this Access Condition List shall succeed if all of the conditions in the List of Access Conditions succeed; otherwise the service shall fail.

f) alternate The service controlled by this Access Condition List shall succeed if any of the conditions in the List of Access Conditions succeed; otherwise the service shall fail.

```
AccessCondition ::= CHOICE {
    never           [0] IMPLICIT NULL,
    semaphore       [1] ObjectName, -- Semaphore Name
    user            [2] CHOICE {
        association     ApplicationReference,
        none            NULL
        },
    password        [3] Authentication-value,
    joint           [4] IMPLICIT SEQUENCE OF AccessCondition,
    alternate       [5] IMPLICIT SEQUENCE OF AccessCondition }
```

9.1.3 MMS Access services

For each of the affected services, the service procedure shall begin with the following steps:

a) If the Access Control List specified by the &accessControl field of the VMD contains an Access Control Element whose Service Class attribute matches the service class of the requested service, the Access Condition of that Access Control Element shall be evaluated. If the Access Condition does not succeed, the service shall fail, returning an error of class ACCESS and an error code of OBJECT-ACCESS-DENIED.

b) If the Access Control List specified by the &accessControl field of the object of the service (if any) contains an Access Control Element whose Service Class attribute matches the service class of the requested service, the Access Condition of that Access Control Element shall be evaluated. If the Access Condition does not succeed, the service shall fail, returning an error of class ACCESS and an error code of OBJECT-ACCESS-DENIED.

c) The remainder of the service procedure shall be performed.

9.1.3.1 Access Condition evaluation

The evaluation of the Access Condition shall be performed as follows:

a) If the Access Condition = NEVER, the Access Condition fails.

b) If the Access Condition = SEMAPHORE, the semaphore specified by the Semaphore Name attribute of the Access Condition shall be examined. If the Application Reference of any owner of the semaphore matches the &client field of the Application Association whose &Transactions field contains the present Transaction object, the Access Condition shall succeed; otherwise it shall fail. If the present Transaction object is contained in the &EATransactions field of the VMD, the Access Condition shall fail.

c) If the Access Condition = USER, the &client field of the Application Association whose &Transactions field contains the present Transaction object shall be compared to the Application Reference attribute of the Access Condition. If they match, the Access Condition shall succeed; otherwise it shall fail. If the present Transaction object is contained in the &EATransactions field of the VMD, the Access Condition shall succeed if the Application Reference attribute of the Access Condition specifies the value NONE for USER; otherwise it shall fail.

d) If the Access Condition = PASSWORD, if the &authenticationValue field of the Application Association object is present, and if &authenticationValue field matches the Password attribute of the Access Condition, the Access Condition shall succeed; otherwise it shall fail. Annex A describes the Authentication-value and the conditions for match. If the present Transaction object is contained in the &EATransactions field of the VMD, the Access Condition shall fail.

e) If the Access Condition = JOINT, the Access Conditions specified in the List of Access Condition attribute shall be evaluated. If all of the Access Conditions in the list succeed, this Access Condition shall succeed; otherwise this Access Condition shall fail. The order of evaluation of the Access Conditions of the List of Access Condition attribute shall be a local matter.

f) If the Access Condition = ALTERNATE, the Access Conditions specified in the List of Access Condition attribute shall be evaluated. If any of the Access Conditions succeed, this Access Condition shall succeed; otherwise this Access Condition shall fail. The order of evaluation of the Access Conditions of the List of Access Condition attribute shall be a local matter.

9.1.4 Reporting services

For those services that return a value of the MMS Deletable parameter, the service shall return a value false if the Access Control List object referenced by the &accessControl field of the object contains any Access Control Element specifying Service Class = DELETE and Access Condition = NEVER. Otherwise, these services shall return a value of true. The value shall not depend on the Access Control List object referenced by the &accessControl field of the VMD. These services include:

a) GetDomainAttributes
b) GetProgramInvocationAttributes
c) GetVariableAccessAttributes
d) GetScatteredAccessAttributes
e) GetNamedVariableListAttributes
f) GetNamedTypeAttributes
g) ReportSemaphoreStatus
h) GetEventConditionAttributes
i) GetEventActionAttributes
j) GetEventEnrollmentAttributes
k) ReportJournalStatus
l) GetDataExchangeAttributes
m) GetAccessControlAttributes

9.2 AccessCondition parameter

The Access Condition parameter is a parameter common to several Access Control services. It expresses a condition required to be satisfied if the service is to be allowed to proceed.

9.2.1 Structure

The structure of the component parameters is shown in Table 15.

Table 15 - Access Condition parameter

Parameter Name	Req/Rsp	Ind/Cnf
Access Condition	M	M(=)
Never	S	S(=)
Semaphore	S	S(=)
User	S	S(=)
Password	S	S(=)
Joint	S	S(=)
List Of Access Condition	M	M(=)
Alternate	S	S(=)
List Of Access Condition	M	M(=)

9.2.1.1 Access Condition

This parameter shall convey the condition to be satisfied. One of the following parameters shall be selected.

9.2.1.1.1 Never

Selection of this parameter shall indicate that the service class identified in this Access Control Element shall always fail, i.e., it shall never succeed.

9.2.1.1.2 Semaphore

Selection of this parameter shall indicate that the service class identified in this Access Control Element shall fail if the MMS client requesting this service does not own the named semaphore. The value of this parameter is the name of the semaphore so identifi ed.

9.2.1.1.3 User

Selection of this parameter shall indicate that the service class identified in the Access Control Element shall fail unless the MMS client requesting this service has the Application Reference identified in this parameter. This parameter may also specify NONE in which case the service shall fail unless the transaction was initiated as an Event Action, i.e., as the result of the occurrence of an event.

9.2.1.1.4 Password

Selection of this parameter shall indicate that the service class identified in this Access Control Element shall fail unless the MMS Client requesting this service has provided the Authentication Value identified in this parameter in the M-ASSOCIATE service.

9.2.1.1.5 Joint

Selection of this parameter shall indicate that this condition shall succeed if all of the Access Conditions in the List of Access Conditions that follow succeed. If this parameter is selected, the following parameter shall appear.

9.2.1.1.5.1 List of Access Condition

This parameter shall contain one or more Access Conditions as described in 9.2.

9.2.1.1.6 Alternate

Selection of this parameter shall indicate that this condition shall succeed if any of the Access Conditions in the List of Access Conditions that follow succeed. If this parameter is selected, the following parameter shall appear.

9.2.1.1.6.1 List of Access Condition

This parameter shall contain one or more Access Conditions as described in 9.2.

9.3 DefineAccessControlList service

The DefineAccessControlList service is used by the MMS client to define a set of conditions that will govern access to MMS objects.

9.3.1 Structure

The structure of the component service primitives is shown in Table 16.

Table 16 - DefineAccessControlList service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Access Control List Name	M	M(=)			
List of Access Control Element	M	M(=)			
Service Class	M	M(=)			
Access Condition	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

9.3.1.1 Argument

This parameter shall convey the parameters of the DefineAccessControlList service request.

9.3.1.1.1 Access Control List Name

This parameter, of type Identifier, shall specify the name of a Access Control List object. Access Control List objects always have VMD-specific scope.

9.3.1.1.2 List of Access Control Element

This parameter shall contain zero or more values describing an access control element. Each such element shall consist of a Service Class parameter indicating which set of MMS services the element constrains, and an Access Condition parameter indicating the condition that, if not satisfied, shall require the services identified by the first parameter to fail.

9.3.1.1.2.1 Service Class

This parameter shall indicate which service class shall be represented in this Access Control Element. The possible values are READ, WRITE, LOAD, STORE, EXECUTE, DELETE and EDIT.

9.3.1.1.2.2 Access Condition

This parameter shall identify the condition that shall be satisfied to allow the service to proceed.

9.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

9.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

9.3.2 Service Procedure

9.3.2.1 Preconditions

The MMS server shall verify that:

a) all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD;

b) no Access Control List object exists with the same name.

If these conditions are not satisfied, the service shall fail and a Result(-) shall be returned.

9.3.2.2 Actions

The MMS server shall create an Access Control List object and assign it attributes as indicated by the parameter list. The &accessControl field shall be initialized to an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose. Each of the Controlled Object fields of the Access Control List object shall be initialized to an empty list. A Result(+) shall be returned.

9.4 GetAccessControlListAttributes service

The GetAccessControlListAttributes service is used by an MMS client to return the fields of the Access Control List object, either the object directly identified, or the object that is referenced by the &accessControl field of the identified object.

9.4.1 Structure

The structure of the component service primitives is shown in Table 17.

Table 17 - GetAccessControlListAttributes service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Access Control List Name	S	S(=)			
VMD	S	S(=)			
Specific Object	S	S(=)			
Object Class	M	M(=)			
Object Name	M	M(=)			
Result(+)			S	S(=)	
Access Control List Name			M	M(=)	
List Of Access Control Element			M	M(=)	
Service Class			M	M(=)	
Access Condition			M	M(=)	
VMD Use			M	M(=)	
Counts of Controlled Objects			C	C(=)	
Object Class			C	C(=)	
Object Count			C	C(=)	
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

9.4.1.1 Argument

This parameter shall convey the parameters of the GetAccessControlListAttributes service request. One of the following parameters shall be chosen.

9.4.1.1.1 Access Control List Name

This parameter, of type Identifier, shall specify the name of a Access Control List whose fields are to be returned. Access Control List objects always have VMD-specific scope.

9.4.1.1.2 VMD

This parameter shall specify that the Access Control List object referenced by the &accessControl field of the VMD is the Access Control List object whose fields are to be returned.

9.4.1.1.3 Specific Object

This parameter shall specify that the Access Control List object referenced by the parameters that follow is the Access Control List object whose fields are to be returned. If this parameter is chosen, the following parameters shall appear.

9.4.1.1.3.1 Object Class

This parameter identifies the object class of the object whose &accessControl field references the Access Control List object to be returned.

9.4.1.1.3.2 Object Name

This parameter identifies the specific object whose &accessControl field references the Access Control List object to be returned.

9.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

9.4.1.2.1 Access Control List Name

This parameter, of type Identifier, shall specify the name of the Access Control List whose fields are being returned. If the Access Control List Name choice was selected for the Argument, this parameter shall have the same value. Access Control List objects always have VMD-specific scope.

9.4.1.2.2 List of Access Control Element

This parameter shall contain zero or more values describing Access Control Elements. Each such element shall consist of a Service Class parameter indicating which set of MMS services the element constrains, and an Access Condition parameter indicating the condition that, if not satisfied, shall require the services identified by the first parameter to fail.

9.4.1.2.2.1 Service Class

This parameter shall indicate which service class is represented in this Access Control Element. The possible values are READ, WRITE, LOAD, STORE, EXECUTE, DELETE and EDIT.

9.4.1.2.2.2 Access Condition

This parameter shall identify the condition to be satisfied to allow the service to proceed.

9.4.1.2.3 VMD Use

This parameter, of type boolean, shall indicate whether (true) or not (false) this Access Control List object is referenced by the &accessControl field of the VMD.

9.4.1.2.4 Counts of Controlled objects

This parameter shall identify the number of objects of each object class that reference this Access Control List object. For each object class the following parameters shall appear. If there are no representatives of a given class in this list, i.e., if the count is zero, these parameters shall not appear.

9.4.1.2.4.1 Object Class

This parameter identifies the object class of the set of objects whose count follows.

9.4.1.2.4.2 Object Count

This parameter, of type integer, is number of objects of the specified object class that appear in the corresponding field of this Access Control List object.

9.4.1.2.5 Access Control List

This parameter, of type Identifier, shall indicate the name of the Access Control List object that controls access to this Access Control List object. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

9.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

9.4.2 Service Procedure

If the Access Control List object does not exist, a Result(-) shall be returned.

The MMS server shall return the fields of the Access Control List object identified in the service request. A Result(+) shall be returned with the Access Control List fields.

9.5 ReportAccessControlledObjects service

The Report Access Controlled Objects service returns a (partial) list of objects that reference a specific Access Control List object.

9.5.1 Structure

The structure of the component service primitives is shown in Table 18.

Table 18 - ReportAccessControlledObjects service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Access Control List Name	M	M(=)			
Object Class	M	M(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List Of Object Name			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

9.5.1.1 Argument

This parameter shall convey the parameters of the ReportAccessControlledObjects service request.

9.5.1.1.1 Access Control List Name

This parameter, of type Identifier, shall specify the name of a Access Control List. Access Control List objects always have VMD-specific scope.

9.5.1.1.2 Object Class

This parameter identifies the object class of the set of objects desired.

9.5.1.1.3 Continue After

This parameter shall be present when the MMS client wishes the list of object names returned by the MMS server to begin with a name other than the (logical) first name in the list. If the value of the Continue After parameter does not match an existing name at the VMD of the class specified, the collating sequence specified in 5.4.2 shall be used by the MMS server to determine the name to start after.

9.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

9.5.1.2.1 List of Object Name

This parameter shall contain the names of objects whose &accessControl field identifies the Access Control List object specified by the Access Control List parameter. The list shall contain zero or more entries. The list of names shall be ordered according to 5.4.2.

9.5.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional ReportAccessControlledObjects requests are necessary to retrieve all of the requested information. If true, more requests are necessary (if the MMS client wishes to retrieve more data). If false, either the List of Object Name contains the last name in the list, or the List of Object Name is empty. This parameter shall be false when the List of Object Name contains zero names.

9.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

9.5.2 Service Procedure

If the Access Control List object does not exist, a Result(-) shall be returned.

The MMS server shall return the list of names of objects whose access control is governed by the Access Control List object identified in the request.

9.6 DeleteAccessControlList service

The DeleteAccessControlList service is used by the MMS client to delete an Access Control List object.

9.6.1 Structure

The structure of the component service primitives is shown in Table 19.

Table 19 - DeleteAccessControlList service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Access Control List Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

9.6.1.1 Argument

This parameter shall convey the parameters of the DeleteAccessControlList service request.

9.6.1.1.1 Access Control List Name

This parameter, of type Identifier, shall specify the name of an Access Control List. Access Control List objects always have VMD-specific scope.

9.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

9.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

9.6.2 Service Procedure

9.6.2.1 Preconditions

The MMS server shall verify that:

a) the Access Control List object identified by the Access Control List Name parameter exists.

b) all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE.

c) all the conditions in the Access Control List object referenced by the &accessControl field of the Access Control List object to be deleted are satisfied for the service class = DELETE.

d) all the fields of the Access Control List object that refer to objects controlled by this Access Control List object are empty.

If these conditions are not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED.

9.6.2.2 Action

The MMS server shall delete the Access Control List object and return a Result(+).

9.7 ChangeAccessControl service

The ChangeAccessControl service is used by the MMS client to change the access control specification for a set of objects of a single object class in the VMD or of the VMD itself. It does this by referencing an Access Control List object that contains the conditions describing the access control required.

9.7.1 Structure

The structure of the component service primitives is shown in Table 20.

Table 20 - ChangeAccessControl service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Scope of Change	M	M(=)			
VMD	S	S(=)			
List of Objects	S	S(=)			
Object Class	M	M(=)			
Object Scope	M	M(=)			
List of Object Name	C	C(=)			
Domain Name	C	C(=)			
Access Control List Name	M	M(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Changed			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Number Changed			M	M(=)	

9.7.1.1 Argument

This parameter shall convey the parameters of the ChangeAccessControl service request. One of the following parameters shall be chosen.

9.7.1.1.1 VMD

Selection of this parameter indicates that the &accessControl field of the VMD is to be changed.

9.7.1.1.2 List of Objects

Selection of this parameter indicates that some set of MMS objects shall have their &accessControl field changed. If this parameter is selected, the following parameters shall appear.

9.7.1.1.3 Object Class

This parameter shall specify the class of the objects whose &accessControl field is to be altered.

9.7.1.1.4 Object Scope

This parameter shall specify the extent of applicability of the change of &accessControl field. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the MMS objects of class Object Class specified by the List of Object Name parameter are to have their &accessControl field changed.

AA-Specific - Specifies that all MMS named objects of class Object Class within the scope of the current application association are to have their &accessControl field changed.

DOMAIN - Specifies that all MMS named objects of class Object Class within the scope of the specified Domain are to have their &accessControl field changed.

VMD - Specifies that all MMS named objects of class Object Class having VMD scope are to have their access control changed.

9.7.1.1.4.1 List of Objects

This parameter shall be specified if Object Scope is SPECIFIC. Otherwise, it shall be omitted. If included, it shall specify the names of the objects whose &accessControl field is to be altered. If the Object Class is Domain, Program Invocation, Journal, Data Exchange, or Access Control List, it shall not specify the **domain-specific** choice of Object Name. If the Object Class is Domain, Program Invocation, Semaphore, Data Exchange, or Access Control List , it shall not specify the **aa-specific** choice of ObjectName.

9.7.1.1.4.2 Domain Name

This parameter, of type Identifier, shall be specified if Object Scope is equal to DOMAIN. Otherwise, it shall be omitted. It provides the name of the Domain that includes all MMS named objects of class Object Class that are to have their &accessControl field changed.

9.7.1.1.5 Access Control List Name

This parameter, of type Identifier, shall specify the name of an Access Control List object. Access Control List objects always have VMD-specific scope.

9.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

9.7.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of MMS named objects that matched the name specification in the service request. If the VMD choice was selected for the argument, this value shall be one.

9.7.1.2.2 Number Changed

This parameter, of type integer, shall indicate the number of MMS named objects that had their &accessControl field changed as a result of executing the service procedure.

NOTE The difference between the Number Matched and Number Changed parameter indicate the number of objects that did not have their &accessControl field changed, either because the access control conditions were not satisfied for this object for Service Class = EDIT, or because there was an attempt to change the Service Class = DELETE condition from NEVER to something else.

9.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

9.7.1.3.1 Number Changed

This parameter, of type integer, shall indicate the number of MMS named objects that had their &accessControl field changed as a result of executing the service procedure.

9.7.2 Service Procedure

9.7.2.1 Preconditions

The MMS server shall verify:

a) that the Access Control List object identified by the Access Control List Name parameter exists;

b) that all the conditions in the Access Control List object referenced by the Reference to Access Control List attribute of the VMD are satisfied for the service class = EDIT.

c) If the List of Objects parameter was chosen in the Argument, for each of the MMS named objects of class Object Class specified in the Argument by the Object Scope, and (in the case of DOMAIN) by the Domain Name parameter, or (in the case of SPECIFIC) by the Names in the List of Object Names parameter, that the object identified exists.

d) If the VMD parameter was chosen for the Argument, that all the conditions in the Access Control List specified by the (current) &accessControl field of the VMD are satisfied for the service class = EDIT (see 9.1.3);

If these conditions are not satisfied, a Result(-) shall be returned and the remainder of this procedure shall be skipped.

9.7.2.2 Procedure for List of Objects

If the List of Objects parameter was chosen in the Argument, the following procedure shall be performed.
For each of the MMS named objects of class Object Class specified in the Argument by the Object Scope, and (in the case of DOMAIN) by the Domain Name parameter, or (in the case of SPECIFIC) by the Names in the List of Object Names parameter, the following steps shall be performed:

a) The MMS server shall verify that all the conditions in the Access Control List specified by the (current) &accessControl field of this object are satisfied for the service class = EDIT (see 9.1.3); otherwise, this object shall not have its &accessControl field changed, nor be included in the count of the number of objects changed.

b) If the &accessControl field of the specified object specifies an Access Control List object containing an Access Control Element that identifies the service class DELETE and the Access Condition NEVER, verify that the Access Control List object identified by the Access Control List Name parameter also contains the service class DELETE and the Access Condition NEVER. Otherwise, this object shall not have its &accessControl field changed, nor be included in the count of the number of objects changed.

c) In the (current) Access Control List object referenced by the &accessControl field of the specified object, delete the reference to this object in the proper Controlled Objects List field.

d) Change the &accessControl field of the specified object to reference the Access Control List object identified by the Access Control List Name parameter.

e) In the (new) Access Control List object referenced by the &accessControl field of the specified object, add a reference to this object in the proper Controlled Objects List field.

After all the objects indicated by the argument parameters have been processed, a Result(+) shall be issued with the values assigned to Number Matched and Number Changed parameters.

If an error occurs in the changing of the access control of any object, a Result(-) shall be issued with the Number Changed parameter indicating the number of objects that correctly had their access control changed. Failure to change access control of an object that did not satisfy condition a) or b) shall not be deemed an error.

NOTE Since every object has access control of some kind, it impossible to "remove" access control from an object. The effect of removing access control can be achieved by changing the Access Control List reference to indicate either M_Deletable or M_NonDeletable as appropriate.

9.7.2.3 Procedure for VMD

Change the &accessControl field of the VMD to reference the Access Control List object identified by the Access Control List Name parameter. Return a Result(+) with a value of Number Changed of one.

NOTE The VMD is not referenced within the Controlled Objects List fields of an Access Control List because its method of identification is different from named MMS objects.

10 VMD Support Services

10.1 Introduction

This clause provides no object models; it specifies the following services.

Status
UnsolicitedStatus
GetNameList
Identify
Rename
GetCapabilityList
VMDStop
VMDReset

10.2 Status Response parameter

The Status Response Parameter is used in several services in this clause. Its structure is shown in Table 21.

Table 21 - Status Response parameter

Parameter Name	Rep	Cnf	CBB
Status Response	M	M(=)	
Logical Status	M	M(=)	
Physical Status	M	M(=)	
Local Detail	U	U(=)	
Operation State	C	C(=)	csr
Extended Status	C	C(=)	csr
Extended Status Mask	U	U(=)	csr
Selected Program Invocation	C	C(=)	csr

10.2.1 Logical Status

This parameter, of type integer, shall convey the value of the &logicalStatus field of the VMD object. The VMD object is defined in clause 7.

10.2.2 Physical Status

This parameter, of type integer, shall convey the value of the &physicalStatus field of the VMD object. The VMD object is defined in clause 7.

10.2.3 Local Detail

This optional parameter, of type bitstring, shall convey the value of the &local-detail field of the VMD object. The VMD object is defined in clause 7.

10.2.4 Operation State

This parameter, of type integer, shall convey the value of the &operationState field of the VMD. This field shall be present only if the **csr** CBB has been negotiated.

10.2.5 Extended Status

This parameter, of type bitstring, shall convey the boolean values of the set of extended attributes:

&safety-Interlocks-Violated,
&any-Resource-Power-On,
&all-Resources-Calibrated,
&local-Control.

These attributes are defined in clause 7. This field shall be present only if the **csr** CBB has been negotiated.

10.2.6 Extended Status Mask

This optional parameter, of type bitstring, shall convey the significance of the corresponding bit in the Extended Status parameter. If a bit in this bitstring is one, the corresponding bit in the Extended Status parameter shall be significant. If a bit in this bitstring is zero, the corresponding bit in the Extended Status parameter shall be ignored. The default value for this parameter is all one's. This field shall be present only if the **csr** CBB has been negotiated, and its use is a user option.

10.2.7 Selected Program Invocation

This parameter shall convey the value of &selected-Program-Invocation field of the VMD. This is the Program Invocation that has been selected to be the Controlling Program Invocation for the system. If no Program Invocation has been selected, this parameter shall have the value NONE. This field shall be present only if the **csr** CBB has been negotiated.

10.3 Status service

The Status service is used by an MMS client to determine the general condition or status of a VMD.

10.3.1 Structure

The structure of the component service primitives is shown in Table 22.

Table 22 - Status service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Extended Derivation	M	M(=)			
Result(+)			S	S(=)	
Status Response			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

10.3.1.1 Argument

This parameter shall convey the parameter of the Status service request.

10.3.1.1.1 Extended Derivation

This parameter, of type boolean, shall indicate which method is to be used to derive the Status response. This parameter is applicable only if the MMS server supports two methods for deriving the Status Response, where one method results in a more extensive derivation of the Status Response. If the value of this parameter is true, the method that results in a extended derivation shall be used. If the value is false, the other method shall be used. If the MMS server only supports one method, the value shall be ignored.

NOTE An example of an extended derivation method is the invocation of a set of self-diagnostic routines.

10.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

10.3.1.2.1 Status Response

This parameter shall convey the information about the status of the VMD.

10.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.3.2 Service Procedure

The MMS server shall perform the Status service by determining the information necessary to create a valid response.

10.4 UnsolicitedStatus service

The UnsolicitedStatus service may be used by an MMS-user to spontaneously report its status.

10.4.1 Structure

The structure of the component service primitives is shown in Table 23.

Table 23 - UnsolicitedStatus service

Parameter Name	Req	Ind	CBB
Argument	M	M(=)	
Status Response	M	M(=)	

The parameters in this service have meaning equivalent to the parameters in the Status response and confirm service primitives.

10.4.2 Service Procedure

An MMS-user that is capable of detecting a change in its own status may, at its option, report this change without receiving a Status request. This is accomplished by requesting the UnsolicitedStatus service. The information in the UnsolicitedStatus.request shall reflect the values of the corresponding attributes of the VMD. A UnsolicitedStatus.request shall not be sent if the peer MMS-user did not indicate support of the UnsolicitedStatus service in the Services Supported parameter received in the Initiate Service.

NOTE The choice of application associations (if more than one exists) on which to send the UnsolicitedStatus Service request is a local matter. All associations, one association, or some group may be selected. The use of this service is functionally equivalent to an Event Notification with an Event Action of the Status Service. The practical difference is that by using the Event Notification method, the conditions under which the service is used are directly visible (and modifiable) using MMS services, while in use of the UnsolicitedStatus, the conditions are a local matter and cannot be determined or changed by the MMS user that is to receive the UnsolicitedStatus information.

10.5 GetNameList service

The GetNameList service may be used by an MMS client to request that a MMS server return the list of or part of the list of object names defined at the VMD.

10.5.1 Structure

The structure of the component service primitives is shown in Table 24.

Table 24 - GetNameList service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Object Class	M	M(=)			
Object Scope	M	M(=)			
Domain Name	C	C(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List of Identifier			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

10.5.1.1 Argument

This parameter shall convey the parameters of the GetNameList service request.

10.5.1.1.1 Object Class

This parameter shall specify the object class of the object name to be returned by the responding MMS-user.

10.5.1.1.2 Object Scope

This parameter shall indicate the scope of the object name list to be returned. The possible values are VMD-specific, Domain-specific, and AA-specific.

10.5.1.1.3 Domain Name

This parameter, of type Identifier, shall specify the name of a Domain if the Domain-specific choice of the Object Scope parameter is chosen. Otherwise, this parameter shall not be present.

10.5.1.1.4 Continue After

This parameter, of type Identifier, shall be present when the MMS client wishes the list of object names returned by the MMS server to begin with a name other than the (logical) first name in the list. It shall be a character string containing the name of an object of class and scope specified in the Extended Object Class, Object Class and Object Scope parameters. If the value of the Continue After parameter does not match an existing name at the VMD of the class and scope specified, the collating sequence specified in 5.4.2 shall be used by the MMS server to determine the name to start after as specified in the service procedure.

10.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

10.5.1.2.1 List Of Identifier

This parameter, of type Identifier, shall contain the names of objects existing at the VMD of the class and scope specified in the Object Class and Object Scope parameters, subject to reduction by the action of the Continue After parameter. The returned list shall contain zero or more entries and shall be sorted according to the collating sequence specified in 5.4.2.

10.5.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional GetNameList requests are necessary to retrieve all of the requested information. If true, more requests are necessary (if the MMS client wishes to retrieve more data). If false, then either the List Of Identifier contains the last name in the list, or the List Of Identifier is empty. This parameter shall be false if the List Of Identifier parameter contains zero names.

10.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.5.2 Service Procedure

The MMS server shall determine which objects have a name attribute value that satisfies the class and scope specified in the Extended Object Class and Object Scope parameters of the indication service primitive. The MMS server shall return this list of object names in the List of Identifier parameter of the response service primitive. The elements of the List of Identifier shall be ordered as determined by the collating sequence specified in 5.4.2.

If the Continue After parameter is not present in the indication service primitive, the List of Identifier shall begin with the first name, as determined by the collating sequence specified in 5.4.2. Otherwise, the List Of Identifier shall begin with the first name after the name specified in the Continue After parameter, using the ordering described above. The More Follows parameter shall be returned with the value determined as described above.

The MMS server shall not issue a Result(-) parameter to indicate that no objects of the requested class and scope exist. Instead, a Result(+) parameter shall be issued with an empty List Of Identifier.

NOTE Repeated usage of this service with the Continue After parameter does not guarantee a list that is consistent in time with any other instance of use of this service. This service returns a segment of the list of object names for which the starting point is based on the Continue After parameter.

10.6 Identify service

The Identify service may be used by an MMS client to obtain identifying information from an MMS server.

10.6.1 Structure

The structure of the component service primitives is shown in Table 25.

Table 25 - Identify service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Result(+)			S	S(=)	
Vendor Name			M	M(=)	
Model Name			M	M(=)	
Revision			M	M(=)	
List of Abstract Sytaxes			C	C(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

10.6.1.1 Argument

There are no service specific parameters of the Identify service request.

10.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

10.6.1.2.1 Vendor Name

This parameter, of type character string, shall convey the value of the &vendorName field of the VMD object. The VMD object is defined in clause 7.

10.6.1.2.2 Model Name

This parameter, of type character string, shall convey the value of the &modelName field of the VMD object. The VMD object is defined in clause 7.

10.6.1.2.3 Revision

This parameter, of type character string, shall convey the value of the &revision field of the VMD object. The VMD object is defined in clause 7.

10.6.1.2.4 List Of Abstract Syntaxes

When included this parameter, of type object identifier, shall convey the value of the &AbstractSyntaxes field of the VMD object. The VMD object is defined in clause 7. This parameter shall not be included if the list is empty.

10.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.6.2 Service Procedure

A response service primitive with the Result(+) parameter shall be issued providing the specified information.

10.7 Rename service

The Rename service may be used by an MMS client in order to request that a MMS server change the identifier of an object to a new identifier.

10.7.1 Structure

The structure of the component service primitives is shown in Table 26.

Table 26 - Rename service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Object Class	M	M(=)			
Current Name	M	M(=)			
New Identifier	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

10.7.1.1 Argument

This parameter shall convey the service specific parameters of the Rename service request.

10.7.1.1.1 Object Class

This parameter shall specify the object class of the object name to be renamed by the responding MMS-user.

10.7.1.1.2 Current Name

This parameter, of type object name, shall specify the object name of the object that is to be renamed.

10.7.1.1.3 New Identifier

This parameter, of type Identifier, shall specify the new identifier part for the referenced object name.

10.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

10.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.7.2 Service Procedure

10.7.2.1 Preconditions

The MMS server shall verify that:

a) an object exists with the type specified in Extended Object Class parameter and the object name specified in the Current Name parameter.

b) any conditions specified for Service Class = EDIT are satisfied in the Access Control List object referenced by the VMD.

c) any conditions specified for Service Class = EDIT are satisfied in the Access Control List object referenced by this object.

d) an object does not exist with the same class type and scope of the Current Name parameter and identifier of the New Identifier parameter.

If any of these conditions are not satisfied, it shall return a Result(-) and skip the remainder of this procedure.

10.7.2.2 Actions

The MMS server shall change the identifier of the specified object to that supplied in the New Identifier parameter and return a Result(+).

NOTE The Rename service is provided to augment commissioning facilities to change the name of objects that are supplied as part of a VMD. Indiscriminate use of the service could cause existing applications to fail if the identifiers of referenced objects are renamed.

10.8 GetCapabilityList service

The GetCapabilityList service is used by an MMS client to request that a MMS server return the list of or part of the list of capabilities defined at the VMD.

10.8.1 Structure

The structure of the component service primitives is shown in Table 27.

Table 27 - GetCapabilityList service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List of Capabilities			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

10.8.1.1 Argument

This parameter shall convey the parameters of the GetCapabilityList service request.

10.8.1.1.1 Continue After

This parameter, of type character string, shall be present if the MMS client wishes the List Of Capabilities returned by the MMS server to begin with a capability other than the first capability in the list. If the Continue After parameter does not match an existing capability of the VMD, the collating sequence specified in 5.4.2 shall be used by the MMS server to determine the capability to start after.

10.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

10.8.1.2.1 List Of Capabilities

This parameter, of type character string, shall convey the value of the &Capabilities field of the VMD object. The VMD object is defined in clause 7. The returned list shall contain zero or more entries and shall be sorted according to the collating sequence specified in 5.4.2.

10.8.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional GetCapabilityList requests are necessary to retrieve more of the requested information. If true, more requests are necessary (if the MMS client wishes to retrieve more data). If false, either the List Of Capabilities contains the last capability in the list, or the List Of Capabilities is empty. The More Follows parameter shall be false if the List Of Capability parameter contains zero capabilities.

10.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.8.2 Service Procedure

The MMS server shall return the &Capabilities field of the VMD. If the Continue After parameter is not present in the indication service primitive, the List Of Capabilities parameter shall begin with the first capability, as determined by the collating sequence specified in 5.4.2 from the &Capabilities field of the VMD. Otherwise, the List Of Capabilities parameter shall begin with the first capability after the capability specified in the Continue After parameter , using the ordering described above. The More Follows parameter shall be returned with the value determined as described above.

10.9 VMDStop service

The VMDStop service is used by an MMS client to stop all control activity at the MMS server and to put the VMD into a state where manual intervention is required.

10.9.1 Structure

The structure of the component service primitives is shown in Table 28.

Table 28 - VMDStop service

Conformance: csr Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Result(+)			S	S(=)	
Result(-) Error Type			S M	S(=) M(=)	

10.9.1.1 Argument

There are no parameters in the argument of this service.

10.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

10.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.9.2 Service procedure

The MMS server shall stop all motion and associated control activity as soon as possible and set the value of the &operationState field to **manualInterventionRequired**. All Program Invocations in the **running** state shall be removed from the **running** state. The state of these Program Invocations is a local matter. The &logicalStatus field shall be set to **no-state-changes-allowed**. These actions and their effect on the attributes of the VMD are summarized in Table 29.

Table 29 - VMD attributes after VMDStop

VMD field	State after Service Performed
&operationState	manual-intervention-required
&localControl	undefined
&logicalState	no-state-changes-allowed
&physicalState	needs-commissioning

10.10 VMDReset service

The VMDReset service is used by an MMS client to put the MMS server into an initialized state. This service also provides the ability for the MMS client to request that the MMS server perform self-diagnostics in the initialization of the VMD. All information relating to &physicalStatus will have been validated as a result of this service.

NOTE In performing an initialization routine or in performing self-diagnostics the MMS server may not be able to maintain the application association. However, the association should be maintained, if possible. It is not the intent of this part of ISO 9506 that the connection to be broken as a result of this service.

10.10.1 Structure

The structure of the component service primitives is shown in Table 30.

Table 30 - VMDReset service

Conformance: csr Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Extended Derivation	M	M(=)			
Result(+)			S	S(=)	
Status Response			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

10.10.1.1 Argument

This parameter contains the parameter of the VMDReset service request.

10.10.1.1.1 Extended Derivation

This parameter, of type boolean, indicates whether (true) or not (false) to perform self-diagnostics as part of the VMDReset service.

10.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

10.10.1.2.1 Status Response

This parameter shall convey the information about the status of the VMD.

10.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

10.10.2 Service procedure

Upon receipt of a VMDReset service indication, the MMS server shall perform the following steps:

a) If local conditions prevent completion of this service, return a Result(-).

b) Delete all Program Invocations and Domains whose &accessControl field allow deletion.

c) Perform a Status Service procedure using the Extended Derivation parameter.

If any step of this procedure fails, the service shall fail and a Result(-) response shall be returned. Otherwise a Result(+) shall be returned containing the Status Response.

11 Domain Management Services

The MMS model of the Virtual Manufacturing Device (VMD) described in clause 7 introduces several abstract elements. This clause describes the services that manage Domains. Domains may be dynamic in nature, coming into existence and being removed from the system either by MMS services or by local action. Services are provided to allow an MMS client to manipulate Domains defined at the MMS server.

11.1 Introduction and Models

This clause provides object models for the following objects:

DOMAIN
ULSM (Upload State Machine)

This clause specifies the following services:

InitiateDownloadSequence	RequestDomainDownload
DownloadSegment	RequestDomainUpload
TerminateDownloadSequence	LoadDomainContent
InitiateUploadSequence	StoreDomainContent
UploadSegment	DeleteDomain
TerminateUploadSequence	GetDomainAttributes

A Domain represents a subset of the capabilities of the VMD used for a specific purpose. The attributes of the Domain are described below, followed by a brief description of the services that operate on the Domain object.

Domains come into existence in one of two ways: (1) a Domain is explicitly created by beginning a download process; (2) a Domain may be created as part of the execution of a Program Invocation or by other local means. Domains may also be predefined to the system, existing prior to the establishment of an MMS context.

11.1.1 The Domain Object Model

This clause introduces the model for the Domain.

```
DOMAIN ::= CLASS {
    &name                   Identifier,
-- shall be unique among the names of all Domains within the VMD
    &Capabilities           MMSString,
    &state                  DomainState,
    &aAssociation           APPLICATION-ASSOCIATION OPTIONAL,
-- This field shall be present if and only if
-- the &state field has a value of
-- loading, complete, incomplete, d1, d2, d3 or d9
    &accessControl          ACCESS-CONTROL-LIST,
    &sharable               BOOLEAN,
    &ProgramInvocations PROGRAM-INVOCATION,
    &uploadsInProgress      INTEGER,
-- The following items reflect the Domain content
-- All the items listed have Domain-specific names.
IF (vnam)
    &NamedVariables         NAMED-VARIABLE,
IF (vlis)
    &NamedVariableLists     NAMED-VARIABLE-LIST,
ENDIF
    &NamedTypes             NAMED-TYPE,
```

```
ENDIF
    &EventConditions        EVENT-CONDITION,
    &EventActions           EVENT-ACTION,
    &EventEnrollments       EVENT-ENROLLMENT,
IF (cspi)
    &EventConditionLists    EVENT-CONDITION-LIST
ENDIF
    }
```

11.1.1.1 &name

The &name field uniquely identifies the Domain within the VMD. The &name shall be a VMD-specific Object Name formed according to the rules for MMS Object Names.

11.1.1.2 &Capabilities

The &Capabilities field is a set of implementation specific parameters necessary to partition the total resources of the VMD for a Domain. The value of elements of this set, represented as character strings, are a local matter.

NOTE The intent of &Capabilities is to convey such parameters as memory allocation, processor assignments, and Input Output bindings.

11.1.1.3 &state

The &state field specifies the state of the Domain. Each Domain may be in one of five states: **loading**, **complete**, **incomplete**, **ready**, or **in-use**. The possible values of the &state field depend on the method of creating the Domain. Prior to its creation, the Domain is non-existent. In order to complete the state table, a non-existent Domain is described as being in the **non-existent** state. The **loading** state is an intermediate state that occurs during the loading process. The Domain enters the **ready** state following a successful Download. The **in-use** state differs from the **ready** state in that one or more Program Invocations have been defined using this Domain. The **complete** state is an intermediate state that occurs after the last DownloadSegment has been received but before the Download Sequence has been terminated. The **incomplete** state is an intermediate state that occurs when a Download Sequence is terminated before the loading process is complete. States **d1-d9** represent intermediate states, i.e., states between a request and its response.

```
DomainState ::= INTEGER {
    non-existent    (0),
    loading         (1),
    ready           (2),
    in-use          (3),
    complete        (4),
    incomplete      (5),
    d1              (7),
    d2              (8),
    d3              (9),
    d4              (10),
    d5              (11),
    d6              (12),
    d7              (13),
    d8              (14),
    d9              (15) }
```

11.1.1.4 &aAssociation

This field identifies the application association over which the Domain is being downloaded. During the process of downloading a Domain, the Domain is dependent on the Application Association over which the Domain was created. If the Application Association is lost before the Domain is placed in the ready state, the Domain shall be deleted. There are no MMS services that report the value of this field.

11.1.1.5 &accessControl

The &accessControl field identifies an Access Control List object that provides conditions under which this Domain may not be uploaded, deleted, or have its access control or name changed.

11.1.1.6 &sharable

The &sharable field specifies whether this Domain may be used in more than one Program Invocation definition at the same time.

NOTE A Domain that is read-only, that is, is not altered by the action of the Program Invocation, is normally sharable. In many cases, Domains that can be modified by execution of a Program Invocation are not sharable; however, by careful coordination a Domain could be modified by two Program Invocations simultaneously, thereby providing a means of inter-process communication. Sharable does not necessarily imply read-only.

11.1.1.7 &ProgramInvocations

This field identifies a set Program Invocations that currently use this Domain. If the Domain is not sharable, this set has at most one Program Invocation. If the Domain is in the **in-use** state, the set shall not be empty.

11.1.1.8 &uploadsInProgress

This field specifies the number of Upload Sequences currently active for this Domain. The value of this field shall be the number of ULSMs that exist for this Domain. The value zero indicates that no upload is currently in progress for this Domain.

11.1.1.9 &NamedVariables

This field identifies the Named Variable objects whose name scope is Domain-specific and which are contained within this Domain. This field is present only if the **vnam** parameter CBB has been negotiated. Named Variables are described in clause 14.

11.1.1.10 &NamedVariableLists

This field identifies the Named Variable List objects whose name scope is Domain-specific and which are contained within this Domain. This field is present only if the **vnam** and the **vlis** parameter CBBs have been negotiated. Named Variable Lists are described in clause 14.

11.1.1.11 &NamedTypes

This field identifies the Named Type objects whose name scope is Domain-specific and which are contained within this Domain. This field is present only if the **vnam** parameter CBB has been negotiated. Named Types are described in clause 14.

11.1.1.12 &EventConditions

This field identifies the Event Condition objects whose name scope is Domain-specific and which are contained within this Domain. Event Conditions are described in clause 19.

11.1.1.13 &EventActions

This field identifies the Event Action objects whose name scope is Domain-specific and which are contained within this Domain. Event Actions are described in clause 20.

11.1.1.14 &EventEnrollments

This field identifies the Event Enrollment objects whose name scope is Domain-specific and which are contained within this Domain. Event Enrollments are described in clause 21.

11.1.1.15 &EventConditionLists

This field identifies the Event Condition List objects whose name scope is Domain-specific and which are contained within this Domain. Event Condition Lists are described in clause 22.

11.1.2 Upload State Machine

This clause introduces the model of an Upload State Machine

```
ULSM ::= CLASS {
    &ulsmID             INTEGER UNIQUE,
-- shall be unique among all ULSM's within this application association
    &domain             DOMAIN,
    &ulsmState          ULState }

ULState ::= INTEGER {
    non-existent        (0),
    uploading           (1),
    uploaded            (2),
    u1                  (3),
    u2                  (4),
    u3                  (5),
    u4                  (6) }
```

11.1.2.1 &ulsmID

This field identifies the upload among all uploads active on this association.

11.1.2.2 &domain

This field identifies the Domain that is being uploaded.

11.1.2.3 &ulsmState

This field identifies the state of the upload.

11.1.3 Domain State Diagrams

In Figure 8, intermediate states (states that exist only between an indication primitive and the response primitive or between a request primitive and a confirm primitive) are indicated by boxes labelled Dn. While these states are transitory, the Domain may be in such a state for some period of time, and may be reported in the same manner as major states.

Domains that are predefined or that come into existence through local means are restricted to the **ready** and **in-use** states of this diagram. The term Program Invocation Count refers to the number of Program Invocations that are currently bound to this Domain. (See clause 12 for information regarding the binding of Program Invocations to Domains.)

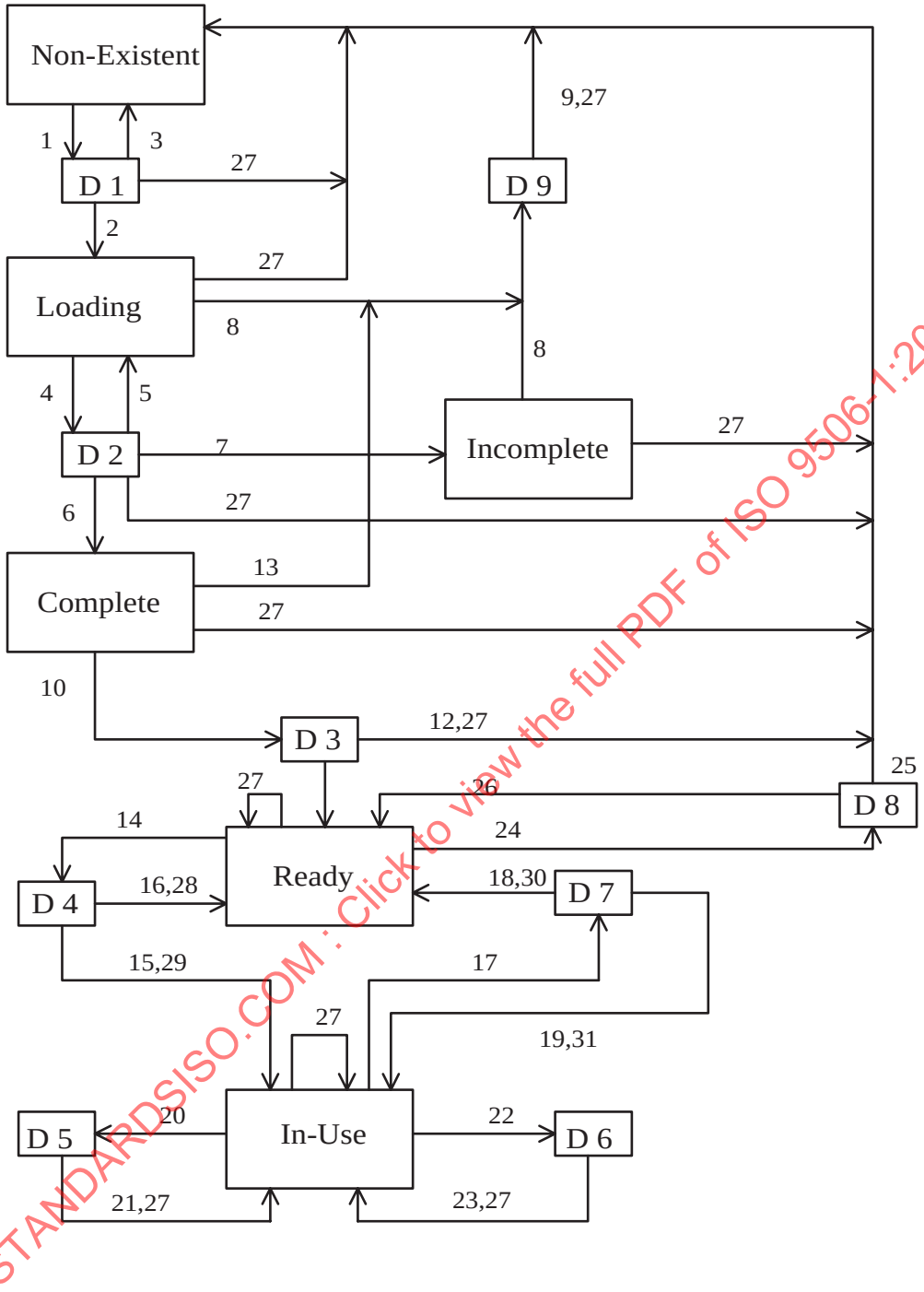

Figure 8 - Domain State Diagram

Transitions of the State Diagram are as follows:

1 - InitiateDownloadSequence.indication
2 - InitiateDownloadSequence.response (+)
3 - InitiateDownloadSequence.response (-)
4 - DownloadSegment.request
5 - DownloadSegment.confirm (+) More Follows = true
6 - DownloadSegment.confirm (+) More Follows = false
7 - DownloadSegment.confirm (-)
8 - TerminateDownloadSequence.request Discard present
9 - TerminateDownloadSequence.confirm (+) or (-)
10 - TerminateDownloadSequence.request Discard not present
11 - TerminateDownloadSequence.confirm (+)
12 - TerminateDownloadSequence.confirm (-)

13 - TerminateDownloadSequence.request Discard present
14 - CreateProgramInvocation.indication
Program Invocation count = 0
15 - CreateProgramInvocation.response (+)
16 - CreateProgramInvocation.response (-)
17 - DeleteProgramInvocation.indication
Program Invocation count = 1
18 - DeleteProgramInvocation.response (+)
19 - DeleteProgramInvocation.response (-)
20 - CreateProgramInvocation.indication
Program Invocation count > 0
21 - CreateProgramInvocation.response (+) or (-)
22 - DeleteProgramInvocation.indication
Program Invocation count > 1
23 - DeleteProgramInvocation.response (+) or (-)
24 - DeleteDomain.indication
25 - DeleteDomain.response (+)
26 - DeleteDomain.response (-)
27 - Abort.indication
28 - Abort.indication Program Invocation creation failed
29 - Abort.indication Program Invocation creation succeeded
30 - Abort.indication Program Invocation deletion succeeded
31 - Abort.indication Program Invocation deletion failed

11.1.4 Segmented Services

There are two sets of services within Domain Management in which the services are required to occur in groups. These are the Download Sequence services and the Upload Sequence services.

11.1.4.1 Download Sequence

The Domain Download Sequence may be used to accomplish the creation and loading of &content of a Domain from the MMS client to the MMS server. Although the MMS client initiates this sequence by requesting the InitiateDownloadSequence Service, subsequent services are controlled by the MMS server. The MMS server shall issue zero or more DownloadSegment Service requests (as required) followed by a request for the TerminateDownloadSequence Service.

Since a Domain may only have one Download Sequence active at any time, the Domain Name is sufficient to identify the Download Sequence being performed. The MMS server shall maintain state information of the Download Sequence as part of the state of the Domain.

If, during the course of a Download Sequence, the association between the client and server is lost, the associated Domain shall be deleted and any partial transfer of information shall be lost. If the association is lost after completion of the Download Sequence, that is, when the Domain is in the **ready** or **in-use** state, the Domain shall be unaffected by the loss of association.

If, during the course of a Download Sequence, any of the service requests is cancelled, the responding MMS-user shall refuse the cancel request unless it is able to maintain the integrity of the Domain. When the processing has progressed to the point that the integrity of the Domain cannot be maintained if cancelled, the &cancelable field of the Transaction object shall be set to false.

11.1.4.2 Upload Sequence

These services may be used to transfer the Domain Content from the MMS server to the MMS client. Upload is accomplished by the MMS client requesting the InitiateUploadSequence service, zero or more UploadSegment services (as required), and a TerminateUploadSequence service, in that order.

If, during the course of an Upload Sequence, the association between the client and server is lost, the Upload Sequence shall be terminated and the associated Upload State Machine (ULSM) shall be deleted. The Domain shall be unaffected. Each successful InitiateUploadSequence service invocation shall create an ULSM that is identified by a unique (among all active ULSMs on the association) ULSM ID. The ULSM shall be created and the ULSM ID assigned at the time of the InitiateUploadSequence by the MMS server. An ULSM may only be referenced via the assigned ULSM ID, and only over the association through which it was assigned. The ULSM shall be deleted and the ULSM ID released via the TerminateUploadSequence service, or when the association is aborted.

If, during the course of an Upload Sequence, any of the service requests is cancelled, the MMS server shall refuse the cancel request unless it is able to maintain the integrity of the Upload State Machine. When the processing has progressed to the point that the integrity of the state machine cannot be maintained if cancelled, the &cancelable field of the Transaction object shall be set to false.

The MMS services for uploading a Domain are interdependent. This interdependence is specified by the Upload State Machines (ULSM) given in Figure 9.

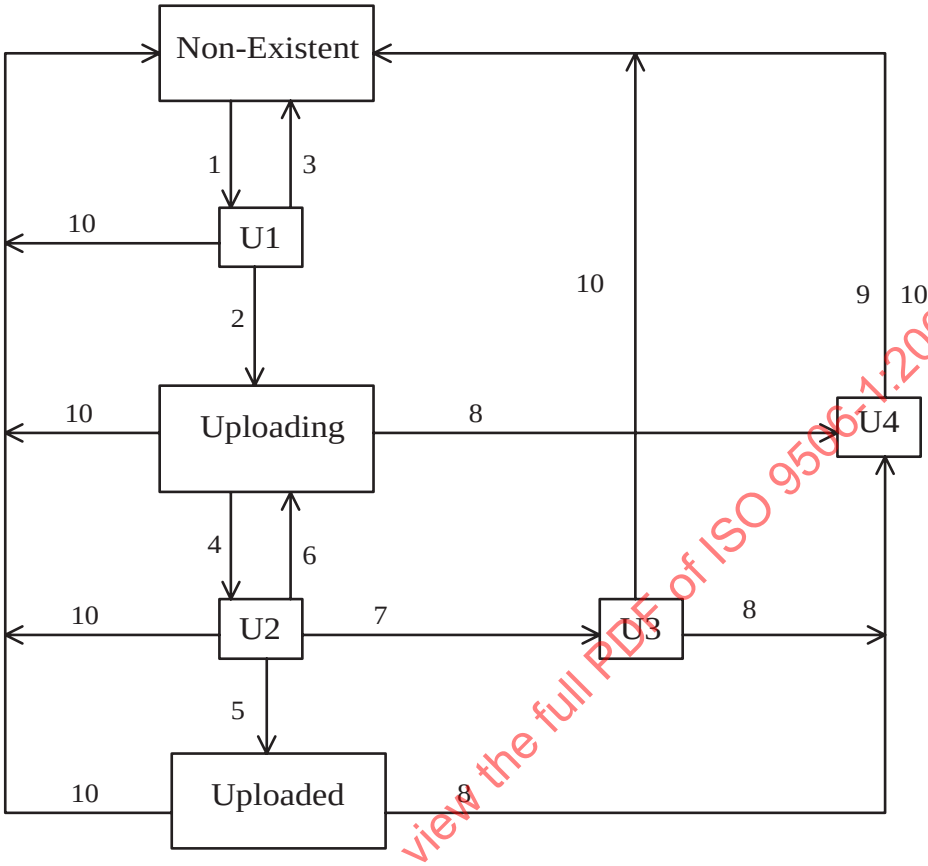

Figure 9 - Upload State Machines

Transitions:

1 - InitiateUploadSequence.indication
2 - InitiateUploadSequence.response(+)
3 - InitiateUploadSequence.response(-)
4 - UploadSegment.indication
5 - UploadSegment.response(+) moreFollows = false
6 - UploadSegment.response(+) moreFollows = true
7 - UploadSegment.response(-)
8 - TerminateUploadSequence.indication
9 - TerminateUploadSequence.response
10 - Abort.indication or Abort.request

11.2 InitiateDownloadSequence service

The InitiateDownloadSequence service is used by the MMS client to request the MMS server to create the named Domain and to begin its loading.

11.2.1 Structure

The structure of the component service primitives is shown in Table 31.

Table 31 - InitiateDownloadSequence service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
List of Capabilities	M	M(=)			
Sharable	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.2.1.1 Argument

This parameter shall convey the parameters of the InitiateDownloadSequence service request.

11.2.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain (at the MMS server) that is to be downloaded.

11.2.1.1.2 List Of Capabilities

This parameter, of type character string, shall represent an implementation specific limitation on the resources of the VMD that are to be part of this Domain. The List of Capabilities becomes a defining element of the Domain. If the List of Capabilities is not valid and available within the resources of the VMD, a Result(-) shall be returned. The determination of valid and available is a local matter.

NOTE The only capabilities that need to be included are those that need be specified in order that the MMS server can properly perform this service request. It is preferable that this parameter not be used at all, since this case promotes the greatest degree of inter-operability. To indicate this situation, a list with zero elements should be specified.

11.2.1.1.3 Sharable

This parameter, of type boolean, shall specify if true that following loading the Domain may be used by more than one Program Invocation concurrently. Such Domains are said to be sharable. The value false shall specify that the Domain may be used by only one Program Invocation.

11.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.2.2 Service Procedure

11.2.2.1 Pre-Conditions

If any of the following conditions is not satisfied, the MMS server shall return a Result(-):

a) The proposed Domain name is not currently in use as a Domain name within the VMD.

b) The list of capabilities is valid and available.

c) The conditions specified for Service Class = LOAD are satisfied in the Access Control List object referenced by the &accessControl field of the VMD.

11.2.2.2 Actions

The MMS server shall create a new Domain and initialize it as follows:

a) The &name field of the newly created Domain shall be the Domain Name parameter of the service indication.

b) The &Capabilities field of the newly created Domain shall be the List of Capabilities parameter of the service indication.

c) The &state field of the newly created Domain shall be **loading**.

d) The &aAssociation field of the newly created Domain shall be set to indicate the association over which the service indication was received.

e) The &accessControl field of the newly created Domain shall be set to an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

f) The &sharable field of the newly created Domain shall be the Sharable parameter of the service indication.

g) The &ProgramInvocations field of the newly created Domain shall be set to an empty set.

h) The &uploadsInProgress field of the newly created Domain shall be zero.

i) The &content field of the newly created Domain shall be initialized to the null object.

The MMS server shall add a reference to the newly created Domain to the &Domains field of the Access Control List object referenced by the &accessControl field of the Domain. The MMS server shall perform any other actions necessary to prepare for the Download Sequence, and shall issue a Result(+) service primitive.

11.3 DownloadSegment service

This service is used by the MMS server to request that a segment of download information be transferred by the MMS client.

NOTE This service is different from most MMS services in that the MMS server issues the request service primitive and receives the confirm service primitive, while the MMS client receives the indication service primitive and issues the response service primitive (see 26.2.1.1).

11.3.1 Structure

The structure of the component service primitives is shown in Table 32.

Table 32 - DownloadSegment service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
Result(+)			S	S(=)	
Load Data			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.3.1.1 Argument

This parameter shall convey the parameter of the DownloadSegment service request.

11.3.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the Domain that is to be loaded. The Domain shall be in the **loading** state.

11.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

11.3.1.2.1 Load Data

This parameter shall contain the information to be downloaded. This parameter shall be either an octet string or an externally encoded value. The MMS server shall use this information to construct the &content field of the Domain. As part of this process, the MMS server shall create and assign values to all the subordinate objects of this Domain. This International Standard does not specify transformation services for this data.

NOTE This International Standard makes no requirements regarding the nature of the information in Load Data. Load Data may have been created as a result of an Upload Sequence (see 11.5) or as a result of a target device specific programming function (such as an APT post-processor).

11.3.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether (true) or not (false) any additional Load Data remains to be transmitted for the Domain named in the Download Sequence. This parameter shall be false if Load Data is a zero length string.

11.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.3.2 Service Procedure

11.3.2.1 Preconditions

The MMS client shall have received a positive response to an InitiateDownloadSequence request naming the Domain identified in the Domain Name parameter of this service indication, and shall not have received a TerminateDownloadSequence indication for this same Domain. If this condition is not satisfied, the MMS client shall return a Result(-) response to the service request.

11.3.2.2 Actions

The MMS client shall prepare Load Data for transmission to the MMS server. The procedure used to create and segment the Load Data shall be a local matter. If there is such Load Data to transmit, it shall become the value of the Load Data parameter and the More Follows parameter shall be set equal to true or false, according to whether there remains more Load Data to be transmitted on a subsequent DownloadSegment service response. If there is no more Load Data to transmit, the MMS client shall set the Load Data Parameter to a zero length string and the More Follows parameter equal to false.

The MMS server shall receive the Load Data segments and interpret the Load Data according to a Domain-specific format, storing it as appropriate. If the More Follows parameter is false, the Domain shall be placed in the **complete** state following completion of this service. If More Follows is true, the Domain shall remain in the **loading** state.

If the MMS client detects an error that invalidates the content of the Domain being downloaded, it shall issue a Result(-) response to the service request. Otherwise, it shall issue a Result(+) response to the service request.

If the MMS server receives a Result(-) confirm as a result of this service request, it shall issue a TerminateDownloadSequence request (see 11.4). If the MMS server receives a Result(+) confirm with the More Follows parameter value of false, it shall issue a TerminateDownloadSequence request (see 11.4). Otherwise it shall issue additional DownloadSegment service requests, as appropriate.

11.4 TerminateDownloadSequence service

The TerminateDownloadSequence service is used by the MMS server to indicate to the MMS client that the Download Sequence is complete.

NOTE This service is different from most MMS services in that the MMS server issues the request service primitive and receives the confirm service primitive, while the MMS client receives the indication service primitive and issues the response service primitive (see 26.2.1.1).

11.4.1 Structure

The structure of the component service primitives is shown in Table 33.

Table 33 - TerminateDownloadSequence service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
Discard	C	C(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.4.1.1 Argument

This parameter shall convey the parameters of the TerminateDownloadSequence service request.

11.4.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the Domain whose Download Sequence is to be terminated. The Domain shall be in the **loading**, **complete**, or **incomplete** state.

11.4.1.1.2 Discard

This parameter, of type ServiceError, shall indicate if present that the downloaded Domain has been deleted. If the Domain has been deleted, this parameter shall provide an indication of the problem encountered. In that case, the Download Sequence shall have been aborted and the MMS server shall have discarded any portion of the Domain Content that has been received. Otherwise, the downloaded Domain Content shall be retained and the Domain shall be placed in the **ready** state.

11.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.4.2 Service Procedure

11.4.2.1 Preconditions

The MMS client shall have received a positive response to an InitiateDownloadSequence request naming the Domain identified in the Domain Name parameter of this service indication, and shall not have received a TerminateDownloadSequence indication for this same Domain. If this condition is not satisfied, the MMS client shall return a Result(-) response to the service request.

11.4.2.2 Actions

If the MMS server detects an unrecoverable error in the course of the Download Sequence, it shall provide the Discard parameter describing the error. If the MMS client has provided a Result(-) to a DownloadSegment request (see 11.3.2), the MMS server shall provide a Discard parameter indicating that the error was detected by the MMS client. If the MMS server has successfully completed the Download Sequence (and therefore has not set the Discard parameter) and the MMS client indicates success by returning a Result(+) to the TerminateDownloadSequence request, the Download Sequence has succeeded. Otherwise the Download Sequence shall have failed.

If the Download Sequence has succeeded, the Domain shall be placed in the **ready** state. If the Download Sequence has failed, any Load Data shall be discarded, the reference to the Domain shall be removed from the Access Control List object referenced by the &accessControl field of the Domain, and the Domain shall be deleted.

11.5 InitiateUploadSequence service

The InitiateUploadSequence service is used by the MMS client to request the MMS server to prepare to upload the Domain of the specified name.

11.5.1 Structure

The structure of the component service primitives is shown in Table 34.

Table 34 - InitiateUploadSequence service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
Result(+)			S	S(=)	
ULSM ID			M	M(=)	
List of Capabilities			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.5.1.1 Argument

This parameter shall convey the parameter of the InitiateUploadSequence service request.

11.5.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the name of the Domain whose content is to be transferred to the MMS client (uploaded).

11.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

11.5.1.2.1 ULSM ID

This parameter, of type integer, shall specify the ULSM object that is created as a result of this request.

11.5.1.2.2 List Of Capabilities

This parameter, of type list of character string, shall identify the &Capabilities that were used in the creation or definition of this Domain.

NOTE The only Capabilities that need to be included are those that must be specified in order that the MMS server can properly perform this service. It is preferable that this parameter not be used at all, since this promotes the greatest degree of inter-operability. To indicate this, a list with zero elements should be specified.

11.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.5.2 Service Procedure

11.5.2.1 Preconditions

The following conditions shall be satisfied:

a) The specified Domain exists and is in the **ready** or the **in-use** state.

b) Any conditions specified for Service Class = STORE are satisfied in the Access Control List object referenced by the &accessControl field of the VMD.

c) Any conditions specified for Service Class = STORE are satisfied in the Access Control List object referenced by the &accessControl field of this Domain.

If any of these conditions are not met, the MMS server shall return a Result(-).

11.5.2.2 Actions

The MMS server shall create a ULSM and assign it a unique integer value. A reference to the new ULSM shall be added to the &Ulsms field of the Application Association. The MMS server shall take whatever other actions are necessary to prepare to upload the specified Domain.

NOTE If an Upload Sequence is undertaken for a Domain that is in the **in-use** state, the &content may be changing during the time the Upload Sequence is being performed. This may result in incomplete or inconsistent data being uploaded. Interpretation of the upload in this situation is a local matter.

11.6 UploadSegment service

The UploadSegment service is used by the MMS client to request the transfer of a segment of upload data from the specified Domain by the MMS server.

11.6.1 Structure

The structure of the component service primitives is shown in Table 35.

Table 35 - UploadSegment service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
ULSM ID	M	M(=)			
Result(+)			S	S(=)	
Load Data			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.6.1.1 Argument

This parameter shall convey the parameter of the UploadSegment service request.

11.6.1.1.1 ULSM ID

This parameter, of type integer, shall specify the instance of the ULSM that controls this transfer. The Domain to be uploaded is implicitly identified by this parameter.

11.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

11.6.1.2.1 Load Data

This parameter shall contain the requested Load Data from the MMS server. This parameter shall be either an octet string or an externally encoded value.

11.6.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether (true) or not (false) more Load Data remains to be transferred to complete the Upload Sequence. This parameter shall be false if Load Data is a zero length string.

11.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.6.2 Service Procedure

11.6.2.1 Preconditions

The Upload State Machine identified by the ULSM ID shall exist. If this condition is not satisfied, a Result(-) shall be returned.

NOTE This condition is equivalent to the statement that a prior InitiateUploadSequence request has been received and a positive response issued, and that a corresponding TerminateUploadSequence request has not been received.

11.6.2.2 Actions

The MMS server shall provide the content of each upload segment such that it is formatted for receipt as Load Data in a later DownloadSegment (download) service. If the Load Data cannot be placed in this format, a Result(-) response shall be returned. If the end of the Load Data has been reached in this sequence, the MMS server shall return More Follows equals false as part of its response.

11.7 TerminateUploadSequence service

The TerminateUploadSequence service is used by the MMS client to request that the MMS server terminate an Upload Sequence. In particular, the TerminateUploadSequence service causes the corresponding ULSM to be deleted, whether or not the service completed or was successful.

11.7.1 Structure

The structure of the component service primitives is shown in Table 36.

Table 36 - TerminateUploadSequence service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
ULSM ID	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.7.1.1 Argument

This parameter shall convey the parameter of the TerminateUploadSequence service request.

11.7.1.1.1 ULSM ID

This parameter, of type integer, shall identify the instance of the ULSM whose Upload Sequence is to be terminated.

11.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.7.2 Service Procedure

11.7.2.1 Preconditions

The Upload State Machine identified by the ULSM ID shall exist. If this condition is not satisfied, a Result(-) shall be returned.

NOTE This condition is equivalent to the statement that a prior InitiateUploadSequence request has been received and a positive response issued, and that a corresponding TerminateUploadSequence request has not been received.

11.7.2.2 Procedure

The MMS client shall request the TerminateUploadSequence service following completion of the Upload Sequence or following an error reported by the MMS server in a Result(-) response to a UploadSegment service. The reference to this ULSM in the &Ulsms field of the Application Association shall be removed. Following either the successful or unsuccessful completion of this service, the ULSM shall be deleted. If the MMS server detects a error in the Upload Sequence, it shall return an Result(-) response. Such a response shall be returned, for instance, if the MMS client requests the TerminateUploadSequence service before the MMS server has returned the More Follows parameter with a value equal to false.

11.8 RequestDomainDownload service

The RequestDomainDownload service is used by the MMS server to request that the MMS client initiate a Download Sequence with the MMS server.

NOTE This service is different from most MMS services in that the MMS server issues the request service primitive and receives the confirm service primitive, while the MMS client receives the indication service primitive and issues the response service primitive (see 26.2.1.1).

11.8.1 Structure

The structure of the component service primitives is shown in Table 37.

Table 37 - RequestDomainDownload service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
List of Capabilities	U	U(=)			
Sharable	M	M(=)			
File Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.8.1.1 Argument

This parameter shall convey the parameters of the RequestDomainDownload service request.

11.8.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the &name field of the Domain that is to be downloaded.

11.8.1.1.2 List Of Capabilities

This optional parameter, of type list of character string, shall be used, if present, as the value of the List of Capabilities parameter in the subsequent Initiate Download Sequence service request.

11.8.1.1.3 Sharable

This parameter, of type boolean, shall indicate if true that the Domain may be used by multiple Program Invocations. Such Domains are said to be sharable. Otherwise, the Domain may be used by only one Program Invocation.

11.8.1.1.4 File Name

This parameter, of type FileName, shall specify the name of the file (as known by the MMS client) containing the information to be loaded.

11.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.8.2 Service Procedure

11.8.2.1 Preconditions

If the file indicated by the File Name parameter does not exist or cannot be accessed, a Result(-) response shall be returned.

11.8.2.2 Actions

Following receipt of the RequestDomainDownload indication service primitive, the MMS client shall request the InitiateDownloadSequence service as described in 11.2. The values of the Domain Name and Sharable parameters received in the indication service primitive shall be used as the values of the parameters of the same name for the InitiateDownloadSequence service. If the List of Capabilities parameter is present in the service indication, this value shall be used as the value of the parameter of the same name for the Initiate Download Sequence service request; otherwise, the value of this parameter of the Initiate Download Sequence service request shall be a local matter. The file identified by the File Name parameter shall be used as the source of the load data.

NOTE 1 If the Domain Content at the MMS client was the result of a previous Domain Upload, the value of the List of Capabilities parameter retained from that upload will normally be used as the value of the List of Capabilities parameter in the Initiate Download Sequence service request.

NOTE 2 A request to cancel a RequestDomainDownload service may require very complex processing if the procedure has progressed to the point where a Domain has been created. It is a local matter when to set the &cancelable field of the RequestDomainDownload Transaction object to false.

Following completion of the TerminateDownloadSequence service, the MMS client shall issue a response to the RequestDomainDownload service. If the load has completed successfully, the Result(+) parameter in the response primitive shall indicate success of the RequestDomainDownload service. If any element of the Download Sequence returns a Result(-) response, that response shall be reflected in a Result(-) response to the RequestDomainDownload service.

11.9 RequestDomainUpload service

The RequestDomainUpload service is used by the MMS server to request that the contents of a specified Domain located at the MMS server be uploaded to the MMS client.

NOTE This service is different from most MMS services in that the MMS server issues the request service primitive and receives the confirm service primitive, while the MMS client receives the indication service primitive and issues the response service primitive (see 26.2.1.1).

11.9.1 Structure

The structure of the component service primitives is shown in Table 38.

Table 38 - RequestDomainUpload service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
File Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.9.1.1 Argument

This parameter shall convey the parameters of the RequestDomainUpload service request.

11.9.1.1.1 Domain Name

This parameter, of type Identifier, shall identify the Domain whose contents are to be uploaded.

11.9.1.1.2 File Name

This parameter, of type FileName, shall specify the file name as known by the MMS client to be used to store the Domain upload.

11.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.9.2 Service Procedure

11.9.2.1 Preconditions

If the file indicated by the File Name parameter does not exist or cannot be accessed, a Result(-) response shall be returned.

11.9.2.2 Actions

The MMS client shall perform the Upload Sequence as described in 11.5 and store the resulting Load Data in the named file. The value of the List of Capabilities shall be retained by the MMS client and associated with the Load Data.

NOTE A request to cancel a RequestDomainUpload service may require very complex processing if the procedure has progressed to the point where an Upload State Machine has been created. It is a local matter when to set the &cancelable field of the RequestDomainUpload Transaction Object to false.

Following completion of the TerminateUploadSequence service, the MMS client shall issue a response to the RequestDomainUpload service. If the upload is completed successfully, the Result(+) parameter in the response primitive shall

indicate success of the RequestDomainUpload service. If any element of the Upload Sequence returns a Result(-) response, that response shall be reflected in a Result(-) response to the RequestDomainUpload service.

11.10 LoadDomainContent service

The LoadDomainContent service is used by the MMS client to request that the MMS server load a file from its own filestore or from a third party into a designated Domain. A typical sequence of operations involving a third party that employs MMS services is shown in Figure 10.

MMS client	MMS server	Third party
LoadDomainContent request	⇒	
	RequestDomainDownload request	⇒
	⇐	InitiateDownloadSeq request
	InitiateDownloadSeq response	⇒
	DownloadSegment request	⇒
	⇐	DownloadSegment response
	...	
		...
	TerminateDownloadSeq request	⇒
	⇐	TerminateDownloadSeq response
	⇐	RequestDomainDownload response
⇐	LoadDomainContent response	

Figure 10 - LoadDomainContent

11.10.1 Structure

The structure of the component service primitives is shown in Table 39.

Table 39 - LoadDomainContent service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
List of Capabilities	U	U(=)			
Sharable	M	M(=)			
File Name	M	M(=)			
Third Party	U	U(=)			tpy
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.10.1.1 Argument

This parameter shall convey the parameters of the LoadDomainContent service request.

11.10.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the &name field of the Domain that is to be loaded.

11.10.1.1.2 List Of Capabilities

This optional parameter, of type list of character string, shall be used if present in the creation of the named Domain.

11.10.1.1.3 Sharable

This parameter, of type boolean, shall indicate if true that the Domain can be used by several Program Invocations concurrently. Otherwise, the Domain can be used by only one Program Invocation.

11.10.1.1.4 File Name

This parameter, of type FileName, shall specify the name of the file containing the information to be loaded.

11.10.1.1.5 Third Party

This parameter, of type ApplicationReference, shall specify the Application Reference of the Application Process through which the named file may be accessed. Support of processing for this parameter is an implementation option that shall be implemented if support for the **tpy** parameter conformance building block is claimed. If it is implemented, its use is a user option. If this parameter is absent, the method of access to this file is a local matter.

11.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.10.2 Service Procedure

11.10.2.1 Preconditions

The following conditions shall be met:

a) no Domain of the name given by the Domain parameter exists;

b) any conditions specified for Service Class = LOAD are satisfied in the Access Control List object referenced by the &accessControl field of the VMD;

c) if a Third Party is specified, the MMS client is able to establish and maintain an association with the Third Party.

If any of these conditions are not met, a Result(-) shall be returned.

11.10.2.2 Actions

The MMS server shall perform the LoadDomainContent Service as follows:

a) create and initialize a Domain following the service procedure described in 11.2.2.

b) if a Third Party is specified, establish an association with that application if none exists; thereafter take appropriate action to cause the named file to be obtained and the named Domain to be loaded.

c) if Third Party is not specified, perform the necessary steps to obtain the file through local means and load it into the specified Domain.

d) if loading is successful, place the Domain in the **ready** state.

NOTE A request to cancel a LoadDomainContent service may require very complex processing if the procedure has progressed to the point where a Domain has been created. It is a local matter when to set the &cancelable field of the LoadDomainContent Transaction Object to false.

If the load is completed successfully, a Result(+) response primitive shall be issued. If the load is not completed successfully, a Result(-) response primitive shall be issued.

11.11 StoreDomainContent service

The StoreDomainContent service is used by an MMS client to request that the contents of a specified Domain at the MMS server be stored in a file in a filestore. "Storing" a Domain requires whatever processing is necessary such that it may be loaded later using the LoadDomainContent service. A typical sequence of operations involving a third party that employs MMS services is shown below in Figure 11.

MMS client	MMS server	Third party
StoreDomainContent request	⇒	
	RequestDomainUpload request	⇒
	⇐	InitiateUploadSeq request
	InitiateUploadSeq response	⇒
	⇐	UploadSegment request
	UploadSegment response	⇒
		...
	...	
	⇐	TerminateUploadSeq request
	TerminateUploadSeq reqsponse	⇒
	⇐	RequestDomainDownload response
⇐	StoreDomainContent response	

Figure 11 - StoreDomainContent

11.11.1 Structure

The structure of the component service primitives is shown in Table 40.

Table 40 - StoreDomainContent service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
File Name	M	M(=)			
Third Party	U	U(=)			tpy
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.11.1.1 Argument

This parameter shall convey the parameters of the StoreDomainContent service request.

11.11.1.1.1 Domain Name

This parameter, of type Identifier, shall identify the Domain whose contents are to be stored to a file.

11.11.1.1.2 File Name

This parameter, of type FileName, shall identify the file in which the Domain is to be stored.

11.11.1.1.3 Third Party

This optional parameter, of type ApplicationReference, shall identify the Third Party on which the filestore resides that is to receive the contents of the named Domain. The presence of this parameter is an implementation option that shall be implemented if support of the **tpy** parameter conformance building block is claimed. If **tpy** is implemented, its use is a user option. If this parameter is absent, the method of storing the file is a local matter.

11.11.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.11.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.11.2 Service Procedure

11.11.2.1 Preconditions

The following conditions shall be met:

a) a Domain of the name given by the Domain parameter exists;

b) any conditions specified for Service Class = STORE are satisfied in the Access Control List object referenced by the &accessControl field of the VMD.

c) any conditions specified for Service Class = STORE are satisfied in the Access Control List object referenced by the &accessControl field of this Domain.

d) If Third Party is specified, the MMS client is able to establish and maintain an association with the Third Party.

If any of these conditions are not met, a Result(-) shall be returned.

11.11.2.2 Actions

The MMS server shall take appropriate action to cause the &content of the Domain to be stored in the indicated file. The value of the List of Capabilities shall be retained and associated with the Load Data.

NOTE A request to cancel a StoreDomainContent service may require very complex processing if the procedure has progressed to the point where an Upload State Machine has been created. It is a local matter when to set the &cancelable field of the StoreDomainContent Transaction Object to false.

If the upload is completed successfully, a Result(+) response shall be issued. If the upload is not completed successfully, the Result(-) response shall be returned.

11.12 DeleteDomain service

The DeleteDomain service is used by an MMS client to request that an MMS server delete the specified Domain.

11.12.1 Structure

The structure of the component service primitives is shown in Table 41.

Table 41 - DeleteDomain service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.12.1.1 Argument

This parameter shall convey the parameter of the DeleteDomain service request.

11.12.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the &name field of the Domain that is to be deleted.

11.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

11.12.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.12.2 Service Procedure

11.12.2.1 Preconditions

The MMS server shall:

a) verify that specified Domain exists;

b) verify that all the conditions in the Access Control List specified by the &accessControlList field of the VMD are satisfied for the service class = DELETE (see 9.1.3);

c) verify that all the conditions in the Access Control List specified by the &accessControlList field of the Domain are satisfied for the service class = DELETE (see 9.1.3);

d) verify that the Domain is in the **ready** state;

NOTE The requirement that the Domain be in the **ready** state is equivalent to the requirement that no Program Invocation be bound to the Domain. Hence, Domains that are currently in use by some Program Invocation cannot be deleted until the Program Invocation is deleted.

e) verify that there are no uploads in progress for this Domain.

If any of these conditions is not met, the MMS server shall return a Result(-).

11.12.2.2 Actions

The MMS server shall delete the designated Domain by performing the following steps.

a) It shall delete all objects subordinate to the Domain regardless of the state of the Access Control List object referenced by these subordinate objects. In conjunction with the deletion of any object the MMS server shall perform any procedure defined for the deletion of that object. For each subordinate object of this Domain, remove the reference to that object from the corresponding field of the referenced Access Control List object.

b) It shall remove any references to this Domain or to its subordinate objects from the attributes of other objects in the VMD. In particular, the Monitored Variable Reference attribute of an Event Condition object that refers to an MMS variable object subordinate to this Domain shall be removed (see 19.1.1).

c) It shall remove the reference to this Domain from the &Domains field of the Access Control List object referenced by the &accessControl field of this Domain.

If the service cannot be performed, a Result(-) response shall be returned. Otherwise, a Result(+) shall be returned.

11.13 GetDomainAttributes service

The GetDomainAttributes service is used by an MMS client to request that an MMS server return the attributes associated with the specified Domain.

11.13.1 Structure

The structure of the component service primitives is shown in Table 42.

Table 42 - GetDomainAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Domain Name	M	M(=)			
Result(+)			S	S(=)	
List of Capabilities			M	M(=)	
State			M	M(=)	
MMS Deletable			M	M(=)	
Sharable			M	M(=)	
List of Program Invocations			M	M(=)	
Upload in Progress			M	M(=)	
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

11.13.1.1 Argument

This parameter shall convey the parameter of the GetDomainAttributes service request.

11.13.1.1.1 Domain Name

This parameter, of type Identifier, shall specify the &name field of the Domain whose attributes are requested.

11.13.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

11.13.1.2.1 List Of Capabilities

This parameter, of type list of character string, shall specify the &Capabilities field of this Domain.

11.13.1.2.2 State

This parameter, of type DomainState, shall specify whether the Domain is in the **loading**, **complete**, **incomplete**, **ready**, **in-use**, or one of the transition states **d1** through **d9**.

11.13.1.2.3 MMS Deletable

Subclause 9.1.4 specifies the value to be returned by this parameter.

11.13.1.2.4 Sharable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Domain may be simultaneously incorporated in more than one Program Invocation.

11.13.1.2.5 List Of Program Invocations

This parameter, of type list of Identifier, shall specify the Program Invocations that are linked to this Domain. If the Domain is not sharable. there shall be at most one such Program Invocation. If the Domain is not in the **in-use** state, in the state **d5**, or in the state **d6**, this list shall be empty.

11.13.1.2.6 Upload In Progress

This parameter, of type integer, shall indicate the number of ULSMs currently active for this Domain. The value zero indicates that no upload is in progress.

11.13.1.2.7 Access Control List

This parameter, of type Identifier, shall indicate the &name field of the Access Control List object that controls access to this Domain. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

11.13.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

11.13.2 Service Procedure

11.13.2.1 Preconditions

The MMS server shall verify that the specified Domain exists. If the Domain does not exist, a Result(-) shall be returned.

11.13.2.2 Actions

The MMS server shall return the values of the attributes of the Domain as parameters of the corresponding Result(+) response.

NOTE The &content field is not included in the response. The names of objects in the contents may be obtained using the GetNameList Service, specifying Domain-specific scope and naming this Domain (see 10.5).

12 Program Invocation Management Services

12.1 Introduction and Models

This clause provides an object model for the following object:

PROGRAM-INVOCATION

This clause specifies the following services:

CreateProgramInvocation
DeleteProgramInvocation
Start
Kill
GetProgramInvocationAttributes
Select
AlterProgramInvocationAttributes
Stop
Resume
Reset
ReconfigureProgramInvocation

The MMS model of the Virtual Manufacturing Device (VMD) described in clause 7 introduces several abstract elements. This clause covers the operations on Program Invocations. Program Invocations may be dynamic in nature, coming into existence and being removed from the system either by MMS services or by local action. Program Invocations may also be predefined within the VMD. Services are provided to allow an MMS client to affect the behaviour of Program Invocations.

A Program Invocation is a dynamic element that most closely corresponds to an execution thread in a multi-tasking environment. It is either predefined or created, either by MMS services or by local action. It is composed of a set of Domains together with control information necessary for its execution. The execution of a Program Invocation can be thought of as a time series of fundamental operations of the underlying device. Further definition of these fundamental operations is a local matter. In this time history of a Program Invocation, there is one distinguished state in which the Program Invocation is ready for execution but has not yet begun execution. This state is referred to as the **idle** state of the Program Invocation. Some Program Invocations also have a distinguished state in which the Program Invocation has "completed". Program Invocations that reach this state have accomplished their purpose and cannot be placed in execution again. This state is called **unrunnable**, and Program Invocations that normally reach this state are called non-reusable. Program Invocations that are reusable will return to the **idle** state following the normal completion of their execution.

12.1.1 The Program Invocation Object Model

This clause introduces the model of a Program Invocation.

```
PROGRAM-INVOCATION ::= CLASS {
      &name                                 Identifier,
-- shall be unique among all Program Invocations
      &programInvocationState               ProgramInvocationState,
      &Domains                              DOMAIN,
      &accessControl                        ACCESS-CONTROL-LIST,
      &reusable                             BOOLEAN,
      &monitor                              BOOLEAN,
-- The following three fields shall all be present if the value of
-- monitor is true.
-- If present, the &name field of each object instance
-- shall have a value equal to the
-- &name field of this instance of the PROGRAM-INVOCATION.
      &eventCondition                       EVENT-CONDITION OPTIONAL,
      &eventAction                          EVENT-ACTION OPTIONAL,
      &eventEnrollment                      EVENT-ENROLLMENT OPTIONAL,
      &executionArgument                    MMSString,
IF (csr)
      &errorCode                            INTEGER,
      &control                              Control-State,
-- The following field shall be present
-- if and only if the value of the &control field is controlled.
      &controlling-Program-Invocation       PROGRAM-INVOCATION,
-- The following two fields shall be present
-- if and only if the value of the &control field is controlling.
      &Controlled-Program-Invocations       PROGRAM-INVOCATION OPTIONAL,
      &program-Location                     MMSString OPTIONAL,
      &running-Mode                         Running-Mode,
-- The following field shall be present
-- if and only if the value of the &running-Mode field is cycle-limited
      &remaining-Cycle-Count                INTEGER OPTIONAL,
-- The following field shall be present
-- if and only if the value of the &running-Mode field is step-limited
      &remaining-Step-Count                 INTEGER OPTIONAL
ENDIF
      }
```

12.1.1.1 &name

The &name field uniquely identifies the Program Invocation within the VMD. The &name shall be a VMD-specific Object Name formed according to the rules for MMS Object Names.

12.1.1.2 &programInvocationState

The &programInvocationState field shall indicate the principal states of the Program Invocation. In order to complete state diagrams, the value **non-existent** is added to this list to describe the condition before a Program Invocation is created.

```
ProgramInvocationState ::= INTEGER {
    non-existent        (0),
    unrunnable          (1),
    idle                (2),
    running             (3),
    stopped             (4),
    starting            (5),
    stopping            (6),
    resuming            (7),
    resetting           (8) }
```

12.1.1.2.1 unrunnable

This state shall denote a condition in which the Program Invocation may no longer be executed, but has not yet been deleted. This state may be reached by the completion of the Program Invocation if the &reusable field is false or through explicit MMS service action or through other local action.

12.1.1.2.2 idle

This state shall denote the condition of a Program Invocation at a time before it is placed in operation.

NOTE If the Program Invocation is implemented through a sequential procedural programming language, this state may correspond to the "beginning of the program".

12.1.1.2.3 running

This state shall denote the condition of a Program Invocation during its execution. Further definition of "execution" is a local matter. However, the **running** state is usually associated with a process that changes the constituent elements of at least one of its subordinate Domains.

12.1.1.2.4 stopped

This state shall denote a condition in which the Program Invocation is at an intermediate state between the onset of execution and completion. However, execution has ceased and the constituent elements of the subordinate Domains are no longer changing due to the action of this Program Invocation.

12.1.1.2.5 starting

This state shall be a transitory state of the Program Invocation between the **idle** and the **running** states. If the Program Invocation is placed in the **running** state through the action of the MMS Start service, this state corresponds to the time between the receipt of the Start service indication primitive and the issuance of the Start service response primitive. While in the **starting** state, the VMD may perform one or more initialization procedures.

12.1.1.2.6 stopping

This state shall be a transitory state of the Program Invocation between the **running** and the **stopped** states. If the Program Invocation is placed in the **stopped** state through the action of the MMS Stop service, this state corresponds to the time between

the receipt of the Stop service indication primitive and the issuance of the Stop service response primitive. While in the **stopping** state, the VMD may perform one or more procedures.

12.1.1.2.7 resuming

This state shall be a transitory state of the Program Invocation between the **stopped** and the **running** states. If the Program Invocation is placed in the **running** state through the action of the MMS Resume service, this state corresponds to the time between the receipt of the Resume service indication primitive and the issuance of the Resume service response primitive. While in the **resuming** state, the VMD may perform one or more procedures.

12.1.1.2.8 resetting

This state shall be a transitory state of the Program Invocation between the **stopped** state and the **idle** state. If the Program Invocation is placed in the **idle** state through the action of the MMS Reset service, this state corresponds to the time between the receipt of the Reset service indication primitive and the issuance of the Reset service response primitive. While in the **resetting** state, the VMD may perform one or more procedures.

12.1.1.3 &Domains

This field shall be a set of Domains that are subordinate to this Program Invocation. The set shall contain at least one Domain.

12.1.1.4 &accessControl

This field identifies an Access Control List object that provides conditions under which this Program Invocation may be executed, deleted, or have its access control changed, or any combination of the above.

12.1.1.5 &reusable

This field shall indicate whether (true) or not (false) the Program Invocation will return to the **idle** state following normal completion of its execution. If false, the Program Invocation will move to the **unrunnable** state following normal execution.

12.1.1.6 &monitor

This field indicates whether or not program monitoring is in effect for this Program Invocation. A Program Invocation that is being monitored uses the Event Management facilities of MMS to inform an enrolled MMS-user whenever the Program Invocation leaves the **running** state. It does this by creating the objects listed below.

12.1.1.7 &eventCondition

If the &monitor field is true, this field shall identify an Event Condition object with the following field values:

a) the &name field shall be equal to the Program Invocation object's &name field;

b) the &ecClass field shall be equal to **monitored**;

c) the &ecState field shall be **idle**;

d) the &enabled field shall be true;

e) the &monitoredVariable field choice shall be **unspecified** (specifying that the Program Invocation is in the **running** state);

f) the &EEnrollment field shall indicate the single defined Event Enrollment object whose definition follows;

g) the &accessControl field shall indicate an Access Control List object that will report the value of MMS Deletable as true. The predefined symbol 'M_Deletable' may be used for this purpose.

h) the &priority field shall be **normalPriority**;

i) the &severity field shall be **normalSeverity**;

j) the &alarmSummaryReports field shall be false;

k) the &evaluationInterval field value shall be a local matter.

12.1.1.8 &eventAction

If the &monitor field is true, this field shall identify an Event Action object with the following field values:

a) the &name field shall be equal to the Program Invocation object's &name field value;

b) the &accessControl field shall indicate an Access Control List object that will report the value of MMS Deletable as true. The predefined symbol 'M_Deletable' may be used for this purpose.

c) the &confirmedServiceRequest field shall indicate the GetProgramInvocationAttributes service;

d) the &modifier field shall be empty;

e) the &eventEnrollment field shall indicate the single defined Event Enrollment whose definition follows.

12.1.1.9 &eventEnrollment

If the &monitor field is true, this field shall identify an Event Enrollment object with the following field values:

a) the &name field shall be equal to the Program Invocation object's &name field value:

b) the &accessControl field shall indicate an Access Control List object that will report the value of MMS Deletable as true. The predefined symbol 'M_Deletable' may be used for this purpose.

c) the &eeClass field value shall be **notification**;

d) the &eventCondition field shall indicate the Event Condition object described above;

e) the &ECTransitions field shall contain the sole element **active-to-idle**;

f) the &aAssociation field value shall specify the association over which the Program Invocation was created;

g) the &notificationLost field shall be false;

h) the &eventAction shall indicate the Event Action described above;

i) the &duration field shall be determined by the corresponding parameter of the CreateProgramInvocation service;

j) the &clientApplication field shall specify the client application that invoked the CreateProgramInvocation service;

k) the &aaRule field value shall be **none**;

l) the &programInvocationState field value shall be **idle**.

12.1.1.10 &executionArgument

This field contains a character string or an externally coded parameter appropriate to the execution of this Program Invocation. It may be set by either the Start service or by the Resume service. It shall initially be set to a empty string when the Program Invocation is created. Subsequent execution of the Program Invocation may change the value of this field.

12.1.1.11 &errorCode

This field, of type integer, shall identify the last recorded error of the Program Invocation execution. This field is present only if the **csr** parameter CBB has been negotiated. A value of zero indicates that no error has been recorded. The meaning of other values is a local matter, as is the method of resetting this field.

12.1.1.12 &control

This field indicates whether (**controlling**) this Program Invocation is intended to be coupled to another Program Invocation as being in control, or whether (**controlled**) this Program Invocation, after coupling, normally receives control information from another Program Invocation (the Controlling Program Invocation). If neither case applies, this field shall have the value **normal**. This field is present only if the **csr** parameter CBB has been negotiated.

When two Program Invocations are coupled in this manner, the service indications for the Start and Resume service requests received by a Controlling Program Invocation can cause state transitions in the Controlled Program Invocations that are coupled to the Controlling Program Invocation (see 12.4 and 12.6). The result reported by a Controlling Program Invocation can contain information that reflects the effect of the service request on the Controlled Program Invocations.

The intent is to associate the execution of a task program that directs the activity of a complex piece of equipment (such as a robot) with a Controlling Program Invocation, and the operation of the hardware control programs of the physical devices with Controlled Program Invocations. This allows logical asynchrony in operation between the two classes of Program Invocations, a necessary feature of the operation of such equipment. In particular, there are now three ways to stop this equipment, by stopping the control program of the equipment itself, by stopping the task program that has the indirect effect of stopping the equipment when no movement commands are generated, and by stopping the entire system (see 12.5).

```
Control-State ::= INTEGER {
    normal          (0),
    controlling     (1),
    controlled      (2) }
```

12.1.1.13 &controlling-Program-Invocation

This field is present only if the **csr** parameter CBB has been negotiated and if the value of the &control field is **controlled**. If present, it references the Program Invocation that acts as the Controlling Program Invocation for this Program Invocation. The referenced Program Invocation shall have its &control field equal to **controlling**, and shall have this Program Invocation included in its &controlled-Program-Invocation field.

12.1.1.14 &Controlled-Program-Invocations

This field is present only if the **csr** parameter CBB has been negotiated and if the Program Invocation has its &control field equal to **controlling**. It is a set of Program Invocations that have their &control field equal to **controlled**, and that have their &controlling-Program-Invocation field indicating the present Program Invocation. This set may be empty.

12.1.1.15 &program-Location

This field, of type character string, is present only if the **csr** parameter CBB has been negotiated and if the Program Invocation has its &control field equal to **controlling**. The use of this field shall be an implementation option, and if used, the format of the &program-Location field shall be described in the CIS (see clause 25 of ISO 9506-2). If present, it identifies the line of source code for the Program Invocation that is currently being executed or will be executed when the Program Invocation is placed in the **running** state.

12.1.1.16 &running-Mode

This field is present only if the **csr** parameter CBB has been negotiated and if the Program Invocation has its &control field equal to **controlling**. If present, it indicates how the Program Invocation execution is governed. If the value of this field is **free-run**, the Program Invocation will stay in the **running** state until some local or remote event occurs that causes it to cease execution. If value of this field is **cycle-limited**, an explicit counter is maintained containing the number of cycles to be executed. When this counter reaches zero, the Program Invocation ceases execution and returns to the **idle** state. This field may take the value **step-**

limited for implementations that support this mode. In that case, the number of steps to be executed shall be specified when the Program Invocation is started or resumed. When the number of steps has been decremented to zero, the Controlling Program Invocation shall automatically move to the **stopped** state. This mode is normally only used for debugging purposes. It shall be indicated in the CSI (see clause 25 of ISO 9506-2) whether the implementation supports the **step-limited** value of &runningMode.

```
Running-Mode ::= INTEGER {
    free-run          (0),
    cycle-limited     (1),
    step-limited      (2) }
```

12.1.1.17 &remaining-Cycle-Count

This field, of type integer, is present only if the **csr** parameter CBB has been negotiated, the value of the &control field is **controlling**, and the value of the &running-Mode field is **cycle-limited**. If present this field is the number of cycles of the Program Invocation remaining to be executed.

12.1.1.18 &remaining-Step-Count

This field, of type integer, is present only if the **csr** parameter CBB has been negotiated, the value of the &control field is **controlling**, and the value of the &running-Mode field is **step-limited**. If present this field is the number of steps remaining to be executed when in Step-Limited Running Mode.

12.1.2 Program Invocation State Diagram

Figure 12 provides the state diagram for Program Invocations.

NOTE In Figure 12, some intermediate states (states that exist only between an indication primitive and a response primitive) may be reported as proper states of the Program Invocation. This is because, in general, the transition may take an appreciable time. The states named "**Pn**" are not reported in the GetProgramInvocationAttributes service since the CreateProgramInvocation and the DeleteProgramInvocation services are defined as atomic (non-interruptible) and therefore appear to be instantaneous to the MMS client. In order to keep the diagram simple, the intermediate states of the Kill service are not shown. These intermediate states are not reported since the Kill service procedure is defined as atomic (non-interruptible), and therefore its effect appears to be instantaneous. The effects of MMS service requests that do not change the state of the Program Invocation, such as GetProgramInvocationAttributes, are not shown on the diagram. The terms destructive and non-destructive are explained in the service procedure for the Cancel service (see 8.5) and in the respective service procedures described in this clause.

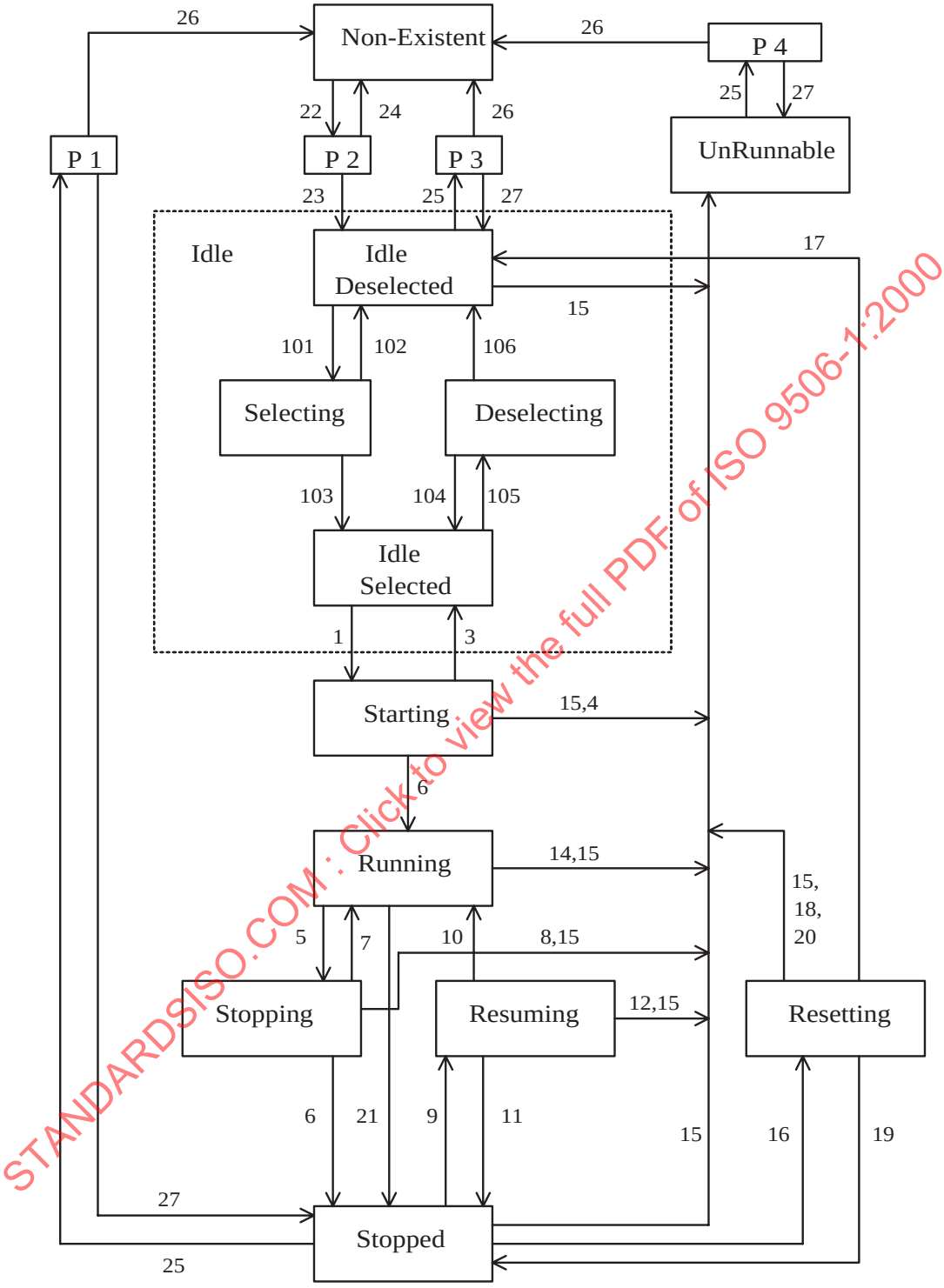

Figure 12 - Program Invocation State Diagram

Transition lines for the model are:

1 - Start.indication
2 - Start.response(+)
3 - Start.response(-) non-destructive
4 - Start.response(-) destructive
5 - Stop.indication
6 - Stop.response(+)
7 - Stop.response(-) non-destructive
8 - Stop.response(-) destructive
9 - Resume.indication
10 - Resume.response(+)
11 - Resume.response(-) non-destructive
12 - Resume.response(-) destructive
13 - (end of program) Reusable = true
14 - (end of program) Reusable = false
15 - Kill.response(+)
16 - Reset.indication
17 - Reset.response(+) Reusable = true
18 - Reset.response(+) Reusable = false
19 - Reset.response(-) non-destructive
20 - Reset.response(-) destructive
21 - (program stop)
22 - CreateProgramInvocation.indication
23 - CreateProgramInvocation.response(+)
24 - CreateProgramInvocation.response(-)
25 - DeleteProgramInvocation.indication
26 - DeleteProgramInvocation.response(+)
27 - DeleteProgramInvocation.response(-)

If **csr** is supported, the following transitions are also recognized:

101 Select.indication
102 Select.response(+)
103 Select.response(-)
104 Select.indication, Controlling Program Invocation is absent
105 Select.response(-)
106 Select.response(+)

If **csr** is not supported, the four states Idle Deselected, Selecting, Deselecting, and Idle Selected together constitute the single state Idle as indicated by the dotted box.

NOTE The transition Kill.indication is not included in this diagram because the action of the Kill service is atomic; there is no difference in the state of the Program Invocation between the indication and the response(+). The Kill.response(-) is not included because it does not cause a state transition.

12.2 CreateProgramInvocation service

The CreateProgramInvocation service is used by an MMS client to assemble Domains into a specified Program Invocation at the VMD. The MMS client specifies a list of Domains that are to be included in the Program Invocation. Note that a given Domain may be in the &Domains field of more than one Program Invocation simultaneously if it is **sharable**.

12.2.1 Structure

The structure of the component service primitive is show in Table 43.

Table 43 - CreateProgramInvocation service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
List of Domain Names	M	M(=)			
Reusable	M	M(=)			
Monitor	U	U(=)			
Monitor Type	C	C(=)			
Control	C	C(=)			csr
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

12.2.1.1 Argument

This parameter shall contain the parameters of the CreateProgramInvocation service request.

12.2.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall be the &name field of this Program Invocation. This name shall be unique among the names of all the Program Invocations in the VMD.

12.2.1.1.2 List Of Domain Names

This parameter, of type list of Identifier, shall specify the existing Domains by name that are to be incorporated as part of this Program Invocation. There shall be at least one such element in this list. The order of the list may be significant to the MMS server in the process of creating the Program Invocation.

12.2.1.1.3 Reusable

This parameter, of type boolean, shall indicate whether (true) the Program Invocation will enter the **idle** state after completion or (false) the **unrunnable** state.

12.2.1.1.4 Monitor

This parameter, of type boolean, shall indicate, if present, that the MMS client wants to be informed about the progress of the Program Invocation as it is executed.

12.2.1.1.5 Monitor Type

This parameter, of type boolean, shall be present if and only if the value of the Monitor parameter is true. This parameter shall indicate, if true, that the notification is **permanent**, hence the notification exists for the life of the Program Invocation. If the value is false, the notification is **current**, and exists only as long as the association is maintained. The value of this parameter shall become the value of the &duration field of the Event Enrollment that is implicitly created as a result of this service request.

12.2.1.1.6 Control

This parameter shall indicate the value of the Control attribute of the Program Invocation. This parameter may have the value **controlling**, **controlled**, or **normal**. This parameter shall be present only if the **csr** parameter CBB has been negotiated.

12.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

12.2.2 Service Procedure

12.2.2.1 Preconditions

The MMS server shall verify that:

a) no Program Invocation of the same name as the Program Invocation Name parameter already exists;

b) all Domains of the List of Domain Names parameter exist;

c) each Domain is available for incorporation into this Program Invocation (that each Domain is either in the **ready** or **d7** state, or is in the **in-use**, **d4**, **d5**, or **d6** state with &sharable field equal to true);

d) any conditions specified for Service Class = LOAD are satisfied in the Access Control List object referenced by the &accessControl field of the VMD;

e) there does not exist an Event Condition, Event Action, or Event Enrollment with the same name of VMD scope as the Program Invocation.

If any of these conditions is not met, a Result(-) shall be returned.

12.2.2.2 Actions

The MMS server shall perform the following series of actions:

12.2.2.2.1 Action Step 1

Create a Program Invocation object initialized as follows:

a) set its &name field equal to the Program Invocation Name parameter;

b) set its &programInvocationState field equal to **idle**;

c) set its &accessControl field to reference an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

d) set the &reusable field equal to the Reusable parameter;

e) set the &monitor field equal to true if the Monitor parameter is specified; otherwise false;

f) set the &executionArgument field equal to a string of length zero.

12.2.2.2.2 Action Step 2

For each Domain in the List of Domain Names parameter, perform the following steps:

a) if the Domain &state field is equal to **d4**, **d5**, or **d6**, wait until the Domain enters the **in-use** state;

b) if the Domain &state field is equal to **d7**, wait until the Domain enters the **ready** state;

c) change the Domain &state field to **in-use**;

d) add a reference to this Domain to the Program Invocation's &Domains field.

e) add a reference to this Program Invocation to the Domain's &ProgramInvocations field.

12.2.2.2.3 Action Step 3

If the Monitor parameter is specified, enable the Program Invocation to report whenever it ceases running. To do this, make use of Event Management facilities described in clause 18. The following actions shall occur that will cause the Program Invocation to issue a EventNotification request primitive that carries a GetProgramInvocationAttribute response whenever the Program Invocation leaves the **running** state (see 12.1.2).
The MMS server shall:

a) create an Event Condition with attributes as described in 12.1.1.7;

b) create an Event Action object with attributes as described in 12.1.1.8;

c) create an Event Enrollment object with attributes as described in 12.1.1.9.

NOTE This is equivalent to transitions of the Program Invocation out of the **running** state, i.e. whenever it completes or is stopped (locally or via MMS services), or is killed if it was in the **running** state.

12.2.2.2.4 Action Step 4

If the Control parameter is present in the service request and has the value CONTROLLING, the &control field shall be set to **controlling**, the &controlled-Program-Invocations field shall be set to an empty list. The &program-Location field shall be set to an empty string and the &running-Mode field shall be set to **free-run**.

If the Control parameter is present in the service request and has the value CONTROLLED, the &control field shall be set to **controlled** and the &controlling-Program-Invocation shall be set to null.

If the Control parameter is present in the service request and has the value NORMAL, the &control field shall be set to **normal**.

12.2.2.3 Conclusion

The MMS server shall guarantee that this service procedure is atomic, (not interruptible by another MMS service indication specifying this Program Invocation or its constitutive elements).

If any step in this process fails, the service shall fail, all partially completed steps shall be undone, and a Result(-) response shall be returned. Otherwise, the service shall succeed, the Program Invocation shall be left in **idle** state, and a Result(+) shall be returned.

12.3 DeleteProgramInvocation service

The DeleteProgramInvocation service is used by an MMS client to cause the deletion of a Program Invocation at the MMS server.

12.3.1 Structure

The structure of the component service primitives is shown in Table 44.

Table 44 – DeleteProgramInvocation service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

12.3.1.1 Argument

This parameter shall contain the parameter of the DeleteProgramInvocation service request.

12.3.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall be the name of the Program Invocation that is to be deleted.

12.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

12.3.2 Service Procedure

12.3.2.1 Preconditions

The MMS server shall:

a) verify that specified Program Invocation exists;

b) verify that all the conditions in the Access Control List referenced by the &accessControl field of the VMD are satisfied for the service class DELETE (see 9.1.3);

c) verify that all the conditions in the Access Control List referenced by the &accessControl field of the Program Invocation are satisfied for the service class DELETE (see 9.1.3);

d) verify that the Program Invocation is in the **idle**, **stopped** or **unrunnable** state;

e) If the &control field of the Program Invocation to be deleted has the value **controlled**, and the value of the &controlling-Program-Invocation field indicates some Program Invocation, verify that the Program Invocation referenced by the &controlling-Program-Invocation field is in the **idle** state.

If these conditions are not met a Result(-) response shall be returned.

12.3.2.2 Actions

The MMS server shall remove references to this Program Invocation from the &ProgramInvocations field of each Domain referenced by the &Domains field of the Program Invocation. If the subject Domain has no other references to Program Invocations, it shall be moved from the **in-use** state to the **ready** state.

If the value of the &monitor field of the Program Invocation object is true, the associated Event Enrollment, Event Action, and Event Condition shall be deleted according to the procedures described in the DeleteEventEnrollment, DeleteEventAction and DeleteEventCondition service procedures, except that any constraints expressed by the Access Control List objects referenced by these objects shall be ignored.

The MMS server shall remove the reference to this Program Invocation of the Access Control List referenced by the &accessControl field of this Program Invocation.

If the &control field of the Program Invocation to be deleted has the value **controlled**, and the value of the &controlling-Program-Invocation field indicates some Program Invocation, alter the &Controlled-Program-Invocations field of the Program Invocation referenced by the &controlling-Program-Invocation field by removing the reference to the Program Invocation about to be deleted.

If the &control field of the Program Invocation to be deleted has the value **controlling**, for each Program Invocation that appears in the &Controlled-Program-Invocations field, perform the following operations:

a) change the value of the &control field of this Program Invocation from **controlled** to **normal**;

b) remove the reference to the Program Invocation to be deleted from the &controlling-Program-Invocation field of this Program Invocation.

The Program Invocation object shall be deleted.

If any step of this process fails, the service shall fail, all partially completed steps shall be undone, and a Result(-) response shall be returned. Otherwise, a Result(+) shall be returned.

12.4 Start service

The Start service is used by an MMS client to change the state of a Program Invocation to the **running** state. The Program Invocation shall be in the **idle** state for this service to be successfully completed.

12.4.1 Structure

The structure of the component service primitives is shown in Table 45.

Table 45 - Start service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Execution Argument	U	U(=)			
Start Location	U	U(=)			csr
Running Mode	C	C(=)			csr
No Limit	S	S(=)			
Cycle Count	S	S(=)			
Step Count	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Program Invocation State			M	M(=)	

12.4.1.1 Argument

This parameter shall contain the parameters of the Start service request.

12.4.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be started.

12.4.1.1.2 Execution Argument

This parameter shall be an optional field that may be used to pass data to the started Program Invocation. This parameter shall be either a character string or an externally coded value.

12.4.1.1.3 Start Location

This optional parameter, of type character string, shall be present only if the Program Invocation has its &control field value equal to **controlling**. If the &control field value is equal to **controlling**, the use of this parameter shall be a user option. If this parameter is implemented, its format shall be (See clause 25 of ISO 9506-2) the same as the &programLocation field of the Program Invocation. The value of this parameter shall indicate the starting location within the Program Invocation at which to begin execution. If the Start Location parameter is omitted from the service request, execution shall begin at the first step of the program. The meaning of the first program step or a default first step is a local matter.

12.4.1.1.4 Running Mode

This parameter shall be present only if the &control field value of the Program Invocation is equal to **controlling**. The value of this parameter shall indicate the value for the &runningMode field of the Program Invocation. Depending on which value of &runningMode field of the Program Invocation object is selected, one of the following parameters shall be present.

12.4.1.1.4.1 No Limit

This parameter, of type null, shall be chosen if the value of the &running-Mode field is to be set to **free-run**.

12.4.1.1.4.2 Cycle Count

This parameter, of type integer, shall be chosen if the value of the &running-Mode field is to be set to **cycle-limited** and the value of the &remaining-Cycle-Count field shall be set to this parameter value. The value of this parameter shall be greater than zero.

12.4.1.1.4.3 Step Count

This parameter, of type integer, shall be chosen if the value of the &running-Mode field is to be set to **step-limited** and the value of the &remaining-Step-Count field shall be set to this parameter value. The value of this parameter shall be greater than zero. Implementation of the **step-limited** &runningMode shall be defined in the CIS (see clause 25 of ISO 9506-2).

12.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

12.4.1.3.1 Program Invocation State

Following an unsuccessful Start service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the **unrunnable** state. This parameter shall identify the state of the Program Invocation following an unsuccessful Start.

12.4.2 Service Procedure

12.4.2.1 Preconditions

The MMS server shall:

a) verify that specified Program Invocation exists;

b) verify that all the conditions in the Access Control List specified by the &accessControl field of the VMD are satisfied for the service class EXECUTE (see 9.1.3);

c) verify that all the conditions in the Access Control List specified by the &accessControl field of the Program Invocation are satisfied for the service class EXECUTE (see 9.1.3);

d) verify that the Program Invocation is in the **idle** state;

e) if the &control field of the Program Invocation identified by the Program Invocation Name parameter of the Start service request does not have the value **controlling**, verify that the Start Location and Running Mode parameters are not present;

f) if the Program Invocation identified by the Program Invocation Name parameter of the Start service request has its &control field equal to **controlling**, verify that the Program Invocation is referenced by the &selected-Program-Invocation field of the VMD.

If any of these conditions is not met, a Result(-) shall be returned.

12.4.2.2 Actions

12.4.2.2.1 Action Step 1

If the Execution Argument parameter is present in the service indication, the &executionArgument field of the Program Invocation object shall have its value set to the value of the Execution Argument parameter; otherwise the value of the &executionArgument field shall remain unchanged.

12.4.2.2.2 Action Step 2

If the **csr** parameter CBB is supported and the &control field of the Program Invocation is equal to **controlling**, the following steps shall be performed for each element of the &Controlled-Program-Invocations field of this Program Invocation:

a) verify that the referenced Program Invocation has its &control field equal to **controlled** and that its &controlling-Program-Invocation field references this Program Invocation;

b) if the referenced Program Invocation is in the **idle** state, perform a Start procedure for this Program Invocation and move it into the **running** state;

c) if the referenced Program Invocation is in the **stopped** state, perform a Resume procedure for this Program Invocation and move it into the **running** state;

d) if the referenced Program Invocation cannot be placed in the **running** state, for every previous referenced Program Invocation of the list, perform a Stop procedure. If any such Stop procedure fails, it is a local matter how to treat this situation. A VMDStop procedure may be the appropriate action. Finally, return a Result(-) for this service request and skip the remainder of this procedure.

12.4.2.2.3 Action Step 3

If the Program Invocation identified by the Program Invocation Name parameter of the Start service request has its &control field equal to **controlling**, the attributes of the Program Invocation shall be set as follows:

a) if the No Limit parameter is selected in the service request, the &running-Mode field of the Program Invocation shall be set to **free-run**;

b) if the Cycle Count parameter is selected in the service request, the &running-Mode field of the Program Invocation shall be set to **cycle-limited** and the &remaining-Cycle-Count field of the Program Invocation shall be set to the value of the Cycle Count parameter;

c) if the Step Count parameter is selected in the service request, the &running-Mode field of the Program Invocation shall be set to **step-limited** and the &remaining-Step-Count field of the Program Invocation shall be set to the value of the Step Count parameter;

d) if the Start Location parameter is present in the service request, the value of this parameter shall condition the program control information of the Program Invocation in order to provide a starting point for the execution of the Program Invocation. The representation used to convey the starting location is a local matter, and in general will depend on the programming language used. The format used for the Start Location parameter shall be described in the CSI (see clause 25 of ISO 9506-2);

e) if the Start Location parameter is absent from the service request, the default value of the starting location shall be used in the execution of the Program Invocation.

12.4.2.2.4 Action Step 4

The Program Invocation shall be placed in the **starting** state. Placing a Program Invocation in the **starting** state means initiating a procedure that will carry the Program Invocation to the **running** state after some time interval (which may be negligible). A Result(+) primitive shall be issued as soon as the Program Invocation is placed in the **running** state. A Result(-) response primitive shall be issued if the starting process fails, and the Program Invocation shall be returned either to the **idle** state if possible or to the **unrunnable** state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation.

Since, in general, starting a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

12.5 Stop service

The Stop service is used by an MMS client to request an MMS server to transition a named Program Invocation from the **running** state to the **stopped** state.

12.5.1 Structure

The structure of the component service primitives is shown in Table 46.

Table 46 - Stop service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Program Invocation State			M	M(=)	

12.5.1.1 Argument

This parameter shall contain the parameters of the Stop service request.

12.5.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be stopped.

12.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

12.5.1.3.1 Program Invocation State

This parameter, of type ProgramInvocationState, shall identify the final state of the Program Invocation if the Stop service fails.

12.5.2 Service Procedure

12.5.2.1 Preconditions

The MMS server shall verify that:

a) the specified Program Invocation exists;

b) all the conditions in the Access Control List specified by the &accessControl field of the VMD are satisfied for the service class EXECUTE (see 9.1.3);

c) all the conditions in the Access Control List specified by the &accessControl field of the Program Invocation are satisfied for the service class EXECUTE (see 9.1.3);

d) the Program Invocation is in the **running** state;

If any of these conditions is not met, a Result(-) shall be returned.

12.5.2.2 Actions

The Program Invocation shall be placed in the **stopping** state. Placing a Program Invocation in the **stopping** state means initiating a procedure that will carry the Program Invocation to the **stopped** state after some time interval (which may be negligible). A Result(+) primitive shall be issued as soon as the Program Invocation is placed in the **stopped** state. A Result(-) response primitive shall be issued if the stopping process fails, and the Program Invocation shall be returned either to the **running** state if possible or to the **unrunnable** state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation.

Since, in general, stopping a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE If the Program Invocation is implemented through a sequential procedural programming language, this service should cause the current program step to be saved as part of the control information of the Program Invocation for a subsequent resumption procedure.

12.6 Resume service

The Resume service is used by an MMS client to request an MMS server to change the state of a Program Invocation to the **running** state.

12.6.1 Structure

The structure of the component service primitives is shown in Table 47.

Table 47 - Resume service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Execution Argument	U	U(=)			
Resume Type	C	C(=)			csr
Continue Mode	S	S(=)			
Change Mode	S	S(=)			
No Limit	S	S(=)			
Cycle Count	S	S(=)			
Step Count	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Program Invocation State			M	M(=)	

12.6.1.1 Argument

This parameter shall contain the parameters of the Resume service request.

12.6.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be resumed.

12.6.1.1.2 Execution Argument

This parameter is an optional field which shall be used to pass data to the resuming Program Invocation. This parameter is either a character string or an externally coded value.

12.6.1.1.3 Resume Type

This parameter shall be present if the **csr** parameter CBB has been negotiated and if the &control field of the Program Invocation is equal to **controlling**. If present, one of the following choices shall be selected.

12.6.1.1.3.1 Continue Mode

This parameter indicates that the Program Invocation is to resume execution with the &running-Mode field unchanged.

12.6.1.1.3.2 Change Mode

This parameter indicates that the Program Invocation is to resume execution with the &running-Mode field changed to a new value. If this parameter is selected, one of the following parameters shall be present.

12.6.1.1.3.2.1 No Limit

This parameter, of type null, shall be chosen if the value of the &running-Mode field is to be changed to **free-run**.

12.6.1.1.3.2.2 Cycle Count

This parameter, of type integer, shall be chosen if the value of the &running-Mode field is to be changed to **cycle-limited**, and the value of the &remaining-Cycle-Count field shall be set to this parameter value. The value of this parameter shall be greater than zero.

12.6.1.1.3.2.3 Step Count

This parameter, of type integer, shall be chosen if the value of the &running-Mode field is to be changed to **step-limited**, and the value of the &remaining-Step-Count field shall be set to this parameter value. The value of this parameter shall be greater than zero. Implementation of the **step-limited** &running-Mode shall be defined in the CSI (see clause 25 of ISO 9506-2). If not supported, this choice may not be selected.

12.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

12.6.1.3.1 Program Invocation State

This parameter shall be of type ProgramInvocationState. Following an unsuccessful Resume service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the **unrunnable** state. This parameter shall identify the state of the Program Invocation following an unsuccessful Resume.

12.6.2 Service Procedure

12.6.2.1 Preconditions

The MMS server shall:

a) verify that specified Program Invocation exists;

b) verify that all the conditions in the Access Control List specified by the &accessControl field of the VMD are satisfied for the service class EXECUTE (see 9.1.3);

c) verify that all the conditions in the Access Control List specified by the &accessControl field of the Program Invocation are satisfied for the service class EXECUTE (see 9.1.3);

d) verify that the Program Invocation is in the **stopped** state;

e) if the &control field of the Program Invocation does not have the value **controlling**, verify that neither the Continue Mode nor the Change Mode parameters are present;

f) if Continue Mode is selected, the MMS server shall:

   1) If the value of the &running-Mode field of the Program Invocation is **cycle-limited**, verify that the value of the &remaining-Cycle-Count field is greater than zero.

2) If the value of the &running-Mode field is **step-limited**, verify that the value of the &remaining-Step-Count field is greater than zero.

3) If the Program Invocation identified by the Program Invocation Name parameter of the Resume service request has its &control field equal to **controlling**, for each element on the &Controlled-Program-Invocations field of this Program Invocation verify that the referenced Program Invocation has its &control field value equal to **controlled** and that its &controlling-Program-Invocation field references this Program Invocation.

If any of these conditions is not met, a Result(-) shall be returned.

12.6.2.2 Actions

a) If the Execution Argument parameter is present, the value of the &executionArgument field of the Program Invocation shall assume the value of the Execution Argument parameter. Otherwise, the value of the &executionArgument field shall remain unchanged.

b) If the Program Invocation identified by the Program Invocation Name parameter of the Resume service request has its &control field equal to **controlling**, the following steps shall be performed for each element on the &Controlled-Program-Invocations field of this Program Invocation:

1) If the referenced Controlled Program Invocation is in the **idle** state, perform a Start procedure for the referenced Controlled Program Invocation and move it into the **running** state.

3) If the referenced Controlled Program Invocation is in the **stopped** state, perform a Resume procedure for the referenced Controlled Program Invocation and move it into the **running** state.

4) If the referenced Controlled Program Invocation cannot be placed in the **running** state, for every previous Controlled Program Invocation of the list, perform a Stop procedure. If any such Stop procedure fails, it is a local matter how to treat this situation. A VMDStop procedure may be the appropriate action. Return a Result(-) for this service request and skip the remainder of this procedure.

c) The Program Invocation shall be placed in the **resuming** state. Placing a Program Invocation in the **resuming** state means initiating a procedure that will carry the Program Invocation to the **running** state after some time interval (which may be negligible). A Result(+) primitive shall be issued as soon as the Program Invocation is placed in the **running** state. A Result(-) response primitive shall be issued if the resuming process fails, and the Program Invocation shall be returned either to the **stopped** state if possible, or to the **unrunnable** state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation.

Since, in general, resuming a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE If the Program Invocation is implemented through a sequential procedural programming language, this service should cause execution of the Program Invocation to continue at the program step indicated in the control information.

12.7 Reset service

The Reset service is used by an MMS client to request an MMS server to transition a named Program Invocation from the **stopped** state to the **idle** state or to the **unrunnable** state, according to the value of the &reusable field of the Program Invocation object.

12.7.1 Structure

The structure of the component service primitives is shown in Table 48.

Table 48 - Reset service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Program Invocation State			M	M(=)	

12.7.1.1 Argument

This parameter shall contain the parameters of the Reset service request.

12.7.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be reset.

12.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

12.7.1.3.1 Program Invocation State

This parameter, of type ProgramInvocationState, shall identify the final state of the Program Invocation if the Reset service fails.

12.7.2 Service Procedure

12.7.2.1 Preconditions

The MMS server shall verify that:

a) the specified Program Invocation exists;

b) all the conditions in the Access Control List specified by the &accessControl field of the VMD are satisfied for the service class EXECUTE (see 9.1.3);

c) all the conditions in the Access Control List specified by the &accessControl field of the Program Invocation are satisfied for the service class EXECUTE (see 9.1.3);

d) the Program Invocation is in the **stopped** state.

If any of these conditions is not met, a Result(-) shall be returned.

12.7.2.2 Actions

The Program Invocation shall be placed in the **resetting** state. Placing a Program Invocation in the **resetting** state means initiating a procedure that will carry the Program Invocation to another state after some time interval (which may be negligible). If the value of the &reusable field is true, the next state shall be the **idle** state; if the value of the &reusable field is false, the next state shall be **unrunnable**. A Result(+) shall be issued as soon as the Program Invocation is placed in the **idle** state or the **unrunnable** state. A Result(-) response primitive shall be issued if the resetting process fails, and the Program Invocation shall be returned to the **stopped** state if possible, or to the **unrunnable** state. The Program Invocation State parameter shall be returned with the Result(-) response to indicate the state of the Program Invocation.

Since, in general, resetting a Program Invocation may take a considerable time, this operation shall be considered cancelable, even if the cancel cannot be done non-destructively. If the service is cancelled, a Result(-) response to the service request shall be returned indicating the resulting state of the Program Invocation, and the cancel service shall issue a Result(+) response.

NOTE If the Program Invocation is implemented through a sequential procedural programming language, this service should cause the control information of the Program Invocation to be altered to reflect that the Program Invocation is again at the "beginning of program".

12.8 Kill service

The Kill service is used by an MMS client to request an MMS server to place a Program Invocation in the **unrunnable** state.

12.8.1 Structure

The structure of the component service primitives is shown in Table 49.

Table 49 - Kill service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

12.8.1.1 Argument

This parameter shall contain the parameter of the Kill service request.

12.8.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall specify the Program Invocation that is to be killed.

12.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

12.8.2 Service Procedure

12.8.2.1 Preconditions

The MMS server shall:

a) verify that specified Program Invocation exists;

b) verify that all the conditions in the Access Control List specified by the &accessControl field of the VMD are satisfied for the service class EXECUTE (see 9.1.3);

c) verify that all the conditions in the Access Control List specified by the &accessControl field of the Program Invocation are satisfied for the service class EXECUTE (see 9.1.3);

If any of these conditions is not met, a Result(-) shall be returned.

12.8.2.2 Actions

The MMS server shall place the named Program Invocation in the **unrunnable** state. If the Kill service fails, the state of the Program Invocation shall be left unchanged.

12.9 GetProgramInvocationAttributes service

The GetProgramInvocationAttributes service is used by an MMS client to request an MMS server to return the attributes associated with the specified Program Invocation.

12.9.1 Structure

The structure of the component service primitives is shown in Table 50.

Table 50 - GetProgramInvocationAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Program Invocation Name	M	M(=)			
Result(+)			S	S(=)	
State			M	M(=)	
List of Domain Names			M	M(=)	
MMS Deletable			M	M(=)	
Reusable			M	M(=)	
Monitor			M	M(=)	
Execution Argument			M	M(=)	
Access Control List			C	C(=)	aco
Error Code			C	C(=)	csr
Control			C	C(=)	csr
Controlling			S	S(=)	
List of Program Invocations			M	M(=)	
Program Location			M	M(=)	
Running Mode			M	M(=)	
Free Running			S	S(=)	
Remaining Cycle Count			S	S(=)	
Remaining Step Count			S	S(=)	
Controlled			S	S(=)	
Controlling Program Invocation			M	M(=)	
Normal			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

12.9.1.1 Argument

This parameter shall contain the parameter of the GetProgramInvocationAttributes service request.

12.9.1.1.1 Program Invocation Name

This parameter, of type Identifier, shall be the name of the Program Invocation whose attributes are requested.

12.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

12.9.1.2.1 State

This parameter, of type ProgramInvocationState, shall be the value of the &programInvocationState field of the Program Invocation.

12.9.1.2.2 List Of Domain Names

This parameter, of type list of Identifier, shall provide the names of the Domains that are referenced by the &Domains field of the Program Invocation.

12.9.1.2.3 MMS Deletable

Subclause 9.1.4 specifies the value to be returned by this parameter.

12.9.1.2.4 Reusable

This parameter, of type boolean, shall be the value of the &reusable field of the Program Invocation.

12.9.1.2.5 Monitor

This parameter, of type boolean, shall be the value of the &monitor field of the Program Inovcation. If this parameter is true, the attributes of the related Event Enrollment object may be obtained using the GetEventEnrollmentAttributes service (see 21.4).

12.9.1.2.6 Execution Argument

This parameter shall contain the value of the &executionArgument field of the Program Invocation. This parameter shall be either a character string or an externally coded value.

12.9.1.2.7 Access Control List

This parameter, of type Identifier, shall be the name of the Access Control List object specified by the &accessControl field of the Program Invocation. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

12.9.1.2.8 Error Code

This parameter, of type integer, shall contain the value of the &errorCode field of the Program Invocation. This parameter shall not appear unless the **csr** parameter CBB has been negotiated.

12.9.1.2.9 Control

This parameter shall contain the &control field of the Program Invocation. This parameter shall not appear unless the **csr** parameter CBB has been negotiated. The &control field may have the value **controlling**, **controlled**, or **normal**. Depending on the value of this field, one of the following parameters shall be present.

12.9.1.2.10 Controlling

This parameter value shall be selected if the value of the &control field of the Program Invocation is **controlling**. If this parameter is selected, the following additional parameters shall appear.

12.9.1.2.11 List of Program Invocations

This parameter shall indicate the names of the Program Invocations in the &Controlled-Program-Invocations field of this Program Invocation. This list may have zero or more elements.

12.9.1.2.12 Program Location

This parameter, of type character string, shall indicate, if present, the value of the &program-Location field if the Program Invocation is related to a sequential programming language. For those Program Invocations that are not related to a sequential programming language, this parameter shall be absent. Use of this parameter shall be described in the CSI (see clause 25 of ISO 9506-2).

12.9.1.2.13 Running Mode

This parameter shall indicate the value of the &running-Mode field of the Program Invocation. Depending on the value of &running-Mode, one of the following parameters shall be present.

12.9.1.2.14 Free Running

Selection of this null parameter shall identify the &running-Mode as **free-run**.

12.9.1.2.15 Remaining Cycle Count

Selection of this parameter shall identify the &running-Mode as **cycle-limited**. Further, the value of the parameter shall be the value of the &remaining-Cycle-Count field of the Program Invocation.

12.9.1.2.16 Remaining Step Count

Selection of this parameter shall identify the &running-Mode as **step-limited**. Further, the value of the parameter shall be the value of the &remaining-Step-Count field of the Program Invocation. Implementation of the step-limited Running Mode shall be defined in the CSI (see clause 25 of ISO 9506-2). If not supported, this choice may not be selected.

12.9.1.2.17 Controlled

This parameter value shall be selected if the value of the &control field of the Program Invocation is **controlled**. If this parameter is selected, the following additional parameter shall appear.

12.9.1.2.18 Controlling Program Invocation

This parameter shall indicate the Program Invocation that is referenced by the value of the &controlling-Program-Invocation field of the Program Invocation. If no such Program Invocation is referenced, this parameter shall have a null value.

12.9.1.2.19 Normal

This parameter value shall be selected if the value of the &control field of the Program Invocation is **normal**. If this parameter is selected, there are no additional parameters.

12.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

12.9.2 Service Procedure

The MMS server shall verify that the specified Program Invocation exists. If the Program Invocation does not exist a Result(-) response shall be returned. Otherwise the attributes of the named Program Invocation shall be returned with a Result(+).

12.10 Select service

The Select service is used by an MMS client to request the MMS server to identify a Program Invocation as the selected Program Invocation controlling the VMD.

12.10.1 Structure

The structure of the component service primitives is shown in Table 51.

Table 51 - Select service

Conformance: csr Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument Controlling Program Invocation List of Program Invocations	M U C	M(=) U(=) C(=)			
Result(+)			S	S(=)	
Result(-) Error Type			S M	S(=) M(=)	

12.10.1.1 Argument

This parameter contains the parameters of the Select service request.

12.10.1.1.1 Controlling Program Invocation

This parameter, of type Identifier, indicates the Program Invocation that is to be selected. The &control field of this Program Invocation shall have a value equal to **controlling**. If this parameter is not present, it indicates that the present Controlling Program Invocation is to be deselected, and no Program Invocation is to be selected.

12.10.1.1.2 List of Program Invocations

This parameter, of type list of identifier, indicates the Program Invocations that will be placed under the control of the referenced Controlling Program Invocation. This parameter shall be present only if the Controlling Program Invocation parameter is present.

12.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

12.10.2 Service procedure

12.10.2.1 Preconditions

The MMS server shall perform the following checks.

a) If the Controlling Program Invocation parameter is present, the Program Invocation indicated by the Controlling Program Invocation parameter shall have its &control field equal to **controlling**.

b) If the Controlling Program Invocation parameter is present, the Program Invocation indicated by the Controlling Program Invocation parameter shall be in the **idle** state.

c) For each element of the List of Program Invocations parameter, verify that the Program Invocation has its &control field equal to **controlled** and that the value of the &controlling-Program-Invocation field is null or equal to the &selected-Program-Invocation field of the VMD.

d) If the &selected-Program-Invocation field of the VMD is not null, verify that this Program Invocation is in the **idle** state.

e) Verify that the &Controlled-Program-Invocations field of the Program Invocation indicated by the Controlling Program Invocation parameter is null.

If any of these conditions is not satisfied, return a Result(-) and skip the remainder of this procedure.

12.10.2.2 Actions

12.10.2.2.1 Action Step 1

If the &selected-Program-Invocation field of the VMD is not null, perform the following steps for the Program Invocation referenced by this field.

a) For each Program Invocation on the &Controlled-Program-Invocations field of the Program Invocation referenced by the Controlling Program Invocation, set the &control field of this Program Invocation to **normal**.

b) Set the &Controlled-Program-Invocations field of the Program Invocation referenced by the &selected-Program-Invocation field of the VMD to an empty list.

c) Set the &selected-Program-Invocation field of the VMD to null.

d) If the Controlling Program Invocation parameter is not present, return Result(+) and skip the remainder of this procedure.

12.10.2.2.2 Action Step 2

a) Set the &selected-Program-Invocation field of the VMD to reference the Program Invocation indicated by the Controlling Program Invocation parameter.

b) If the List of Program Invocations parameter is empty, set the &Controlled-Program-Invocations field of the Program Invocation indicated by the Controlling Program Invocation to an empty list.

c) If the List of Program Invocations parameter is not empty, for each element of the list, set the &controlling-Program-Invocation field of this Program Invocation to reference the Program Invocation indicated by the Controlling Program Invocation parameter. Add this Program Invocation to the &Controlled-Program-Invocations field of the Program Invocation indicated by the Controlling Program Invocation parameter.

d) Return a Result(+).

12.11 AlterProgramInvocationAttributes service

The AlterProgramInvocationAttributes service is used by an MMS client to alter some attributes of a Program Invocation at the MMS server.

12.11.1 Structure

The structure of the component service primitives is shown in Table 52.

Table 52 - AlterProgramInvocationAttributes service

Conformance: csr Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Controlling Program Invocation	M	M(=)			
Running Mode	C	C(=)			
No Limit	S	S(=)			
Cycle Count	S	S(=)			
Step Count	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

12.11.1.1 Argument

This parameter shall contain the parameters of the AlterProgramInvocationAttributes service request.

12.11.1.1.1 Controlling Program Invocation

This parameter, of type Identifier, shall identify the Program Invocation whose attributes are to be altered.

12.11.1.1.2 Running Mode

This parameter shall indicate the value for the &running-Mode field of the Program Invocation object. Depending on the selection of a value for the &running-Mode, one of the following parameters shall be present.

12.11.1.1.2.1 No Limit

This parameter shall be chosen if the value of the &running-Mode field is to be set to **free-run**.

12.11.1.1.2.2 Cycle Count

This parameter, of type integer, shall be chosen if the value of the &running-Mode field is to be set to **cycle-limited**; the value of the &remaining-Cycle-Count field shall be set to this parameter value.

12.11.1.1.2.3 Step Count

This parameter, of type integer, shall be chosen if the value of the &running-Mode field is to be set to **step-limited**; the value of the &remaining-Step-Count field shall be set to this parameter value. Implementation of the **step-limited** value of the &running-Mode field shall be defined in the CSI (see clause 25 of ISO 9506-2). If not supported, this choice may not be selected.

12.11.2 Service procedure

12.11.2.1 Preconditions

The MMS server shall verify that:

a) the Program Invocation identified by the Controlling Program Invocation parameter exists; and

b) the Program Invocation has its &control field equal to **controlling**.

If either condition is not satisfied, return a Result(-) and skip this procedure.

12.11.2.2 Actions

The MMS server shall:

a) If the No Limit parameter is selected in the service request, the &running-Mode field of the Program Invocation shall be set to **free-run**.

b) If the Cycle Count parameter is selected in the service request, the &running-Mode field of the Program Invocation shall be set to **cycle-limited** and the &remaining-Cycle-Count field of the Program Invocation shall be set to the value of the Cycle Count parameter.

c) If the Step Count parameter is selected in the service request, the &running-Mode field of the Program Invocation shall be set to **step-limited** and the &remaining-Step-Count field of the Program Invocation shall be set to the value of the Step Count parameter.

12.12 ReconfigureProgramInvocation service

The ReconfigureProgramInvocation service is used by an MMS client to add and/or remove Domains to/from a Program Invocation.

12.12.1 Structure

The structure of the component service primitives is shown in Table 53.

Table 53 - ReconfigureProgramInvocation service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Old Program Invocation Name	M	M(=)			
New Program Invocation Name	U	U(=)			
List of Domains to Add	M	M(=)			
List of Domains to Remove	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

12.12.1.1 Argument

This parameter shall contain the parameters of the ReconfigureProgramInvocation service request.

12.12.1.1.1 Old Program Invocation Name

This parameter, of type Identifier, shall be the name of the Program Invocation that is to be reconfigured.

12.12.1.1.2 New Program Invocation Name

This optional parameter, of type Identifier, provides the possibility to rename the Program Invocation in the same operation. If this name is already in use, a Result(-) response shall be returned.

12.12.1.1.3 List of Domains to Add

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be added to the &Domains field of the Program Invocation object.

12.12.1.1.4 List of Domains to Remove

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be removed from the &Domains field of the Program Invocation object.

12.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

12.12.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

12.12.2 Service Procedure

12.12.2.1 Preconditions

The MMS server shall

a) verify that the Program Invocation identified by the Old Program Invocation parameter exists;

b) verify that this Program Invocation is in **running** state;

c) if the New Program Invocation parameter is present, verify that no Program Invocation having a **&name** field equal to the New Program Invocation parameter already exists.

d) verify the existence of all Domains of the List of Domains to Add list;

e) verify the existence of all Domains of the List of Domains to Remove list;

f) verify that each Domain on the List of Domains to Add parameter is available for incorporation into this Program Invocation (that each Domain is either in the **ready** or **d7** state, or is in the **in-use**, **d4**, **d5**, or **d6** state with **&sharable** field equal to true);

If any of these conditions is not satisfied, return a Result(-), and skip the remainder of this procedure.

12.12.2.2 Actions

The MMS server shall:

a) for each Domain on the List of Domains to Add parameter, perform the following steps:

   1) if the **&state** field of the Domain is equal to **d4**, **d5**, or **d6**, wait until the Domain enters the **in-use** state;

   2) if the &state field of the Domain is equal to **d7**, wait until the Domain enters the **ready** state;

3) change the value of the &state field of the Domain to **in-use**;

4) add this Program Invocation to the &ProgramInvocations field of the Domain;

5) add this Domain to the &Domains field of the Program Invocation.

b) for each Domain on the List of Domains to Remove parameter, perform the following steps:

1) remove the Program Invocation from the &ProgramInvocations field of the Domain;

2) if the &ProgramInvocations field is empty, change the value of the &state field of the Domain to **ready**;

3) remove the Domain from the &Domains field of the Program Invocation.

c) If the New Program Invocation Name parameter is present, change the &name field of the Program Invocation to this parameter.

d) Return a Result(+).

The MMS server shall guarantee that this service procedure is atomic, (not interruptible by another MMS service indication specifying this Program Invocation or its constitutive elements).

13 Unit Control

13.1 Introduction and Models

This clause provides an object model for the following object:

UNIT-CONTROL

This clause specifies the following services:

InitiateUnitControlLoad	AddToUnitControl
UnitControlLoadSegment	RemoveFromUnitControl
UnitControlUpload	GetUnitControlAttributes
StartUnitControl	LoadUnitControlFromFile
StopUnitControl	StoreUnitControlToFile
CreateUnitControl	DeleteUnitControl

The Unit Control object represents a collection of MMS objects, Domains and Program Invocations, that may be loaded and managed as a unit. Downloads and uploads may be performed on the Domains represented by a Unit Control object through a single sequence of operations.

NOTE The Unit Control object may be used to minimize the number of PDU's necessary to download large numbers of Domains or to create large numbers of Program Invocations. Rules concerning the applicability of grouping of these objects is a local matter and outside the scope of this part of ISO 9506.

13.1.1 Unit Control Object Model

```
UNIT-CONTROL ::= CLASS {
    &name                   Identifier,
-- Shall be unique within the VMD
    &accessControl          ACCESS-CONTROL-LIST,
    &Domains                DOMAIN,
    &ProgramInvocations     PROGRAM-INVOCATION }
```


13.1.1.1 &name

This field shall uniquely identify the Unit Control object at the VMD. The name scope of the Unit Control object shall be VMD-specific.

13.1.1.2 &accessControl

This field shall indicate an Access Control List object that governs whether or not the Unit Control object may be deleted using the DeleteUnitControl service.

13.1.1.3 &Domains

This field shall identify the Domain objects that are constituents of the Unit Control object, and that may be affected by operations on the Unit Control object.

13.1.1.4 &ProgramInvocations

This field shall identify the Program Invocation objects that are constituents of the Unit Control object, and that may be affected by operations on the Unit Control object.

13.2 Control Element

The Control Element is a complex parameter used in several services to describe a single element of a Unit Control object.

13.2.1 Structure

The structure of the Control Element parameter is shown in Table 54.

Table 54 - Control Element parameter

Parameter name	Rsp	Cnf
Control Element	M	M(=)
Begin Domain Definition	S	S(=)
Domain Name	M	M(=)
List of Capabilities	M	M(=)
Sharable	M	M(=)
Load Data	M	M(=)
Continue Domain Definition	S	S(=)
Domain Name	M	M(=)
Load Data	M	M(=)
End Domain Definition	S	S(=)
Domain Name	M	M(=)
Program Invocation Definition	S	S(=)
Program Invocation Name	M	M(=)
List of Domains	M	M(=)
Reusable	M	M(=)
Monitor	U	U(=)
Monitor Type	C	C(=)
Program Invocation State	C	C(=)

13.2.1.1 Begin Domain Definition

Selection of this parameter shall indicate that a new Domain is about to be defined. If selected, the following parameters shall appear.

13.2.1.1.1 Domain Name

This parameter, of type Identifier, shall identify the Domain to be processed.

13.2.1.1.2 List of Capabilities

This parameter, of type list of character string, shall identify the &Capabilities associated with this Domain.

13.2.1.1.3 Sharable

This parameter, of type boolean, shall identify the &sharable field of the Domain.

13.2.1.1.4 Load Data

This parameter shall be the initial partial &content or the total &content of the Domain.

13.2.1.2 Continue Domain Definition

Selection of this parameter shall indicate that a Domain identified in a previous Control Element is about to have more data associated with it. If selected, the following parameters shall appear.

13.2.1.2.1 Domain Name

This parameter, of type Identifier, shall identify the Domain to be processed.

13.2.1.2.2 Load Data

This parameter shall be the (partial) &content of the Domain.

13.2.1.3 End Domain Definition

Selection of this parameter shall indicate that a Domain identified in a previous Control Element is now complete. If selected, the following parameter shall appear.

13.2.1.3.1 Domain Name

This parameter, of type Identifier, shall identify the Domain to be processed.

13.2.1.4 Program Invocation Definition

Selection of this parameter shall indicate that a Program Invocation definition follows. If selected, the following parameters shall appear.

13.2.1.4.1 Program Invocation Name

This parameter, of type Identifier, shall indicate the Program Invocation to be defined.

13.2.1.4.2 List of Domains

This parameter, of type list of Identifier, shall indicate the Domains identified in the &Domains field of the Program Invocation.

13.2.1.4.3 Reusable

This parameter, of type boolean, shall identify the &reusable field of the Program Invocation.

13.2.1.4.4 Monitor

This parameter, of type boolean, shall identify the &monitor field of the Program Invocation.

13.2.1.4.4.1 Monitor Type

This parameter, of type boolean, shall be present if and only if the value of the Monitor parameter is true. The use of this parameter is defined in 12.2.1.1.5.

13.2.1.4.5 Program Invocation State

This parameter, of type integer, shall indicate, if present, the value of the &programInvocationState field of the Program Invocation. When used with the UnitControlLoad service, it shall indicate the state into which the Program Invocation shall be placed. When used with the UnitControlUpload service, it shall indicate the actual state of the Program Invocation.

13.3 InitiateUnitControlLoad service

The InitiateUnitControlLoad service is used by an MMS client to request a MMS server to create a Unit Control object and prepare it for loading. The loading process uses two confirmed MMS services such that the MMS server requests elements of the contents of the Unit Control object. Table 55 shows the sequence of the service primitives.

Table 55 - Interaction of Unit Control Primitives

MMS Client	MMS Server
InitiateUnitControlLoad.req -->	
	<-- UnitControlLoadSegment.req
UnitControlLoadSegment.rsp -->	
...	...
	<-- UnitControlLoadSegment.req
UnitControlLoadSegment.rsp -->	
	<-- InitiateUnitControlLoad.rsp

13.3.1 Structure

The structure of the component service primitives of the InitiateUnitControlLoad service is shown in table 56.

Table 56 - InitiateUnitControlLoad service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Domain Name			S	S(=)	
Program Invocation Name			S	S(=)	

13.3.1.1 Argument

This parameter shall convey the parameters of the InitiateUnitControlLoad service request.

13.3.1.1.1 Unit Control Name

This parameter, of type Identifier, shall specify the name of the Unit Control object.

13.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure. When failure is indicated, one of the following parameters shall be returned.

13.3.1.3.1 Domain Name

This parameter shall indicate the Domain that was being created when the error was detected. Either this parameter or the Program Invocation Name parameter shall be selected.

13.3.1.3.2 Program Invocation Name

This parameter shall indicate the Program Invocation that was being created when the error was detected. Either this parameter or the Domain Name parameter shall be selected.

13.3.2 Service procedure

13.3.2.1 Preconditions

If a Unit Control object of the specified name already exists, the MMS server shall return a Result(-).

13.3.2.2 Actions

The MMS server shall issue one or more UnitControlLoadSegment requests, as appropriate, until it receives a response in which the More Follows parameter is false. It shall perform the service procedure prescribed for that service. If, during the processing of the information contained in the response to the UnitControlLoadSegment service request, it detects an error either in creating

a Domain or a Program Invocation, the MMS server shall halt the loading process and return a Result(-) indicating the Domain or Program Invocation for which loading was in progress when the error occurred.

If no error occurs in the processing of the UnitControlLoadSegment services, the MMS server shall return a Result(+).

13.4 UnitControlLoadSegment service

The UnitControlLoadSegment service is used by an MMS server to obtain Load Data elements from the MMS client.

13.4.1 Structure

The structure of the component service primitives is shown in Table 57.

Table 57 - UnitControlLoadSegment service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
List of Control Elements			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.4.1.1 Argument

This parameter shall convey the parameters of the UnitControlLoadSegment service request.

13.4.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD that is to be loaded.

13.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

13.4.1.2.1 List of Control Elements

This parameter shall contain the information necessary to construct the constituent Domains and Program Invocations of the Unit Control object. The presence of the Program Invocation State parameter of each Control Element shall be a user option.

13.4.1.2.2 More Follows

This boolean parameter shall indicate whether (true) or not (false) more UnitControlSegment service requests are needed to complete the construction of the Unit Control object.

13.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.4.2 Service procedure

The MMS server shall issue a UnitControlLoadSegment request specifying the name of the Unit Control object to be loaded. Upon receipt of the response, the MMS server shall perform the following actions.

13.4.2.1 Preconditions

For each item in the List of Control Elements the MMS server shall perform the following actions.

a) If the Control Element specifies the beginning of a Domain Definition, the MMS server shall verify that no Domain of that name exists in the VMD.

b) If the Control Element specifies the continuation of a Domain Definition, the MMS server shall verify that the Domain exists and is in the **loading** state.

c) If the Control Element specifies the end of a Domain Definition, the MMS server shall verify that the Domain exists and that it is in the **loading** state.

d) If the Control Element specifies a Program Invocation definition, the MMS server shall verify that all the Domains in the &Domains field of this Program Invocation exist and that they are in the **ready** state or that they are in the **in-use** state and their &sharable field is true.

13.4.2.2 Actions

For each item in the List of Control Elements the MMS server shall perform the following actions.

a) If the Control Element specifies the beginning of a Domain Definition, the MMS server shall create the Domain, using the List of Capabilities parameter provided, and place it in the **loading** state. If the Load Data parameter is provided, it shall begin the loading process, using the Load Data.

b) If the Control Element specifies the continuation of a Domain Definition, the MMS server shall continue the loading process, using the Load Data parameter provided.

c) If the Control Element specifies the end of a Domain Definition, the MMS server shall place the Domain in the **ready** state.

d) If the Control Element specifies a Program Invocation definition, the MMS server shall create the named Program Invocation, linking it to the indicated Domains. It shall place each of the Domains in the **in-use** state. If the Program Invocation State parameter is present, the MMS server shall place the Program Invocation in the state indicated by this parameter; otherwise, it shall place the Program Invocation in the **idle** state.

NOTE If the association is lost during the course of a sequence of UnitControlLoadSegment services such that a Domain is in an intermediate state, the provisions of 11.1.4.1 apply. Also note the restrictions that clause 8.3 of ISO 9506-2 places on the use of the Conclude service.

13.5 UnitControlUpload service

The UnitControlUpload service is used by an MMS client to obtain Load Data elements from the MMS server. This service may have to be invoked several times to obtain a complete upload of the Unit Control object.

13.5.1 Structure

The structure of the component service primitives is shown in Table 58.

Table 58 - UnitControlUpload service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Continue After	U	U(=)			
Domain Name	S	S(=)			
Upload ID	S	S(=)			
Program Invocation	S	S(=)			
Result(+)			S	S(=)	
List of Control Elements			M	M(=)	
Next Element			C	C(=)	
Domain Name			S	S(=)	
Upload ID			S	S(=)	
Program Invocation			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.5.1.1 Argument

This parameter shall convey the parameters of the UnitControlUpload service request.

13.5.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD that is to be uploaded.

13.5.1.1.2 Continue After

This optional parameter shall indicate where in the list of constituents of the Unit Control object to begin the next Control Element. If this parameter is not present, the upload shall begin at the beginning of the Unit Control object. If this parameter is present, one of the following parameters shall be selected.

13.5.1.1.2.1 Domain Name

This parameter, of type Identifier, shall indicate the next Domain that is to be uploaded.

13.5.1.1.2.2 Upload ID

This parameter, of type integer, shall indicate the upload state machine currently open for some Domain upload that is to be continued.

13.5.1.1.2.3 Program Invocation

This parameter, of type Identifier, shall indicate the next Program Invocation whose definition is to be uploaded.

13.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

13.5.1.2.1 List of Control Elements

This parameter shall contain the information necessary to construct the constituent Domains and Program Invocations of the Unit Control object. The Program Invocation State parameter shall be included in each Program Invocation definition within a Control Element, and its value shall be set corresponding to the &programInvocationState field of the Program Invocation.

13.5.1.2.2 Next Element

This optional parameter shall indicate, if present, the first element not transmitted in this list of Control Elements that should be the next element to be transmitted if another UnitControlUpload request is received. If this parameter is absent, this shall indicate that uploading of the Unit Control object is complete with this service response. If this parameter is present, one of the following parameters shall be selected.

13.5.1.2.2.1 Domain Name

This parameter, of type Identifier, shall indicate the next Domain that is to be uploaded.

13.5.1.2.2.2 Upload ID

This parameter, of type integer, shall indicate the upload state machine currently open for some Domain upload that is to be continued.

13.5.1.2.2.3 Program Invocation

This parameter, of type Identifier, shall indicate the next Program Invocation whose definition is to be uploaded.

13.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.5.2 Service procedure

For the purposes of responding to the UnitControlUpload service request, the MMS server shall maintain the constituents of a Unit Control object in an ordered list. An ordering of Domain Names based on the collating sequence as prescribed in 5.4.2 followed by a similar ordering of Program Invocations is suggested but not required. This part of ISO 9506 requires only that the ordering algorithm used be unambiguous and be such that any Program Invocation appear later in the ordering than any Domains on which it depends.

The MMS client may issue a UnitControlUpload request indicating a position in this ordering by identifying the next Domain to be uploaded, the Domain whose upload is partially complete, or the next Program Invocation definition to be uploaded. If the MMS client does not specify any such element, the upload is to start from the beginning of the list.

The MMS server shall verify the consistency of the &Domains field and the &ProgramInvocations field of the Unit Control object. If any objects so referenced do not exist, the MMS server shall amend the respective list.

The MMS server shall provide definitions for each constituent element of the Unit Control object in order, determined by its ordering algorithm. For each Domain in the &Domains field of the Unit Control object, the MMS server shall create an Upload State Machine (see 11.1.4.2) and transmit all or part of the Domain content. The determination of the necessity of segmentation

and the size of the segments shall be a local matter. For each Program Invocation in the &ProgramInvocations field of the Unit Control object, the MMS server shall transmit a Program Invocation definition record.

If, for any reason, the entire content of the Unit Control object cannot be contained within a single service response, the MMS server shall provide an indication of the next element in the order that has not yet been transmitted.

a) If the next element to be transmitted is a Domain, the MMS server shall indicate the name of this Domain.

b) If a Domain content has been partially transmitted and more content of that Domain remains to be transmitted, the MMS server shall indicate the identify of the Upload State Machine currently active.

c) If the next element to be transmitted is a Program Invocation, the MMS server shall indicate the name of this Program Invocation.

If the present transmission exhausts the Unit Control object, that is, it transmits the last element on the list, the MMS server shall omit the Next Element parameter.

13.6 StartUnitControl service

This service is used by an MMS client to request an MMS server to place all the constituent Program Invocations of a Unit Control object into the **running** state.

13.6.1 Structure

The structure of the component service primitives is shown in Table 59.

Table 59 - StartUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Execution Argument	U	U(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Start Unit Control Error			C	C(=)	
Program Invocation Name			M	M(=)	
Program Invocation State			M	M(=)	

13.6.1.1 Argument

This parameter conveys the parameters of the StartUnitControl service.

13.6.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object whose constituent Program Invocations are to be started.

13.6.1.1.2 Execution Argument

This parameter may be used to pass information to the Program Invocations that are to be started.

13.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure. When failure is indicated, the following parameters may be returned.

13.6.1.3.1 Start Unit Control Error

This parameter shall be included in the Result(-) if the failure was due to the failure of the derived Start procedure on a specific Program Invocation. If this parameter is included, the following fields shall also appear.

13.6.1.3.1.1 Program Invocation Name

This parameter shall indicate the name of the Program Invocation whose Start service failed.

13.6.1.3.1.2 Program Invocation State

This parameter shall indicate the resulting state of the Program Invocation whose Start service has failed. Following an unsuccessful Start service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the **unrunnable** state.

13.6.2 Service procedure

The MMS server shall perform the following actions.

a) For each entry on the &ProgramInvocations field of the Unit Control object, verify that the Program Invocation exists. If a Program Invocation does not exist, remove its reference from the &ProgramInvocations field.

b) For each entry on the &ProgramInvocations field of the Unit Control object, place the Program Invocation in the **running** state. This shall be done as follows:

   1) If the Program Invocation is already in the **running**, **starting**, or **resuming** state, do nothing.

   2) If the Program Invocation is in the **idle** or **resetting** state, perform a Start service procedure (see 12.4).

   3) If the Program Invocation is in the **stopped** or **stopping** state, perform a Resume procedure (see 12.6).

   4) If the Program Invocation is in the **unrunnable** state, return a Result(-) with a Start Unit Control Error parameter indicating the failed Program Invocation and its state.

   5) If any Start procedure on a constituent Program Invocation fails, return a Result(-) with a Start Unit Control Error parameter indicating the failed Program Invocation and its state.

c) Return a Result(+).

13.7 StopUnitControl service

This service is used by an MMS client to request an MMS server to move all the constituent Program Invocations of a Unit Control object into the **stopped** state.

13.7.1 Structure

The structure of the component service primitives is shown in Table 60.

Table 60 - StopUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Stop Unit Control Error			C	C(=)	
Program Invocation Name			M	M(=)	
Program Invocation State			M	M(=)	

13.7.1.1 Argument

This parameter shall convey the parameter of the StopUnitControl service.

13.7.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object whose constituent Program Invocations are to be stopped.

13.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure. When failure is indicated, the following parameters may be returned.

13.7.1.3.1 Stop Unit Control Error

This parameter shall be included in the Result(-) if the failure was due to the failure of the derived Stop procedure on a specific Program Invocation. If this parameter is included, the following fields shall also appear.

13.7.1.3.1.1 Program Invocation Name

This parameter shall indicate the name of the Program Invocation whose Stop service failed.

13.7.1.3.1.2 Program Invocation State

This parameter shall indicate the resulting state of the Program invocation whose Stop service has failed. Following an unsuccessful Stop service, the Program Invocation shall be returned to its previous state if possible, or it shall be placed in the **unrunnable** state.

13.7.2 Service procedure

The MMS server shall perform the following actions.

a) For each element of the &ProgramInvocations field of the Unit Control object, verify that the Program Invocation exists. If this condition is not satisfied, remove the reference to the Program Invocation from the &ProgramInvocations field.

b) For each element of the &ProgramInvocations field of the Unit Control object, place each Program Invocation that is in the **running** state into the **stopped** state. This shall be done as follows:

   1) If the Program Invocation is already in the **stopped**, **stopping**, **idle**, **resetting**, or **unrunnable** state, do nothing.

   2) If the Program Invocation is in the **running** or **starting** state, perform a Stop procedure (see 12.5).

   3) If any Stop procedure on a constituent Program Invocation fails, return a Result(-) with a Stop Unit Control Error parameter indicating the failed Program Invocation and its state.

c) Return a Result(+).

13.8 CreateUnitControl service

The CreateUnitControl service is used by an MMS client to request an MMS server to create a new Unit Control object with a specified set of Domains and/or Program Invocations.

13.8.1 Structure

The structure of the component service primitives is shown in Table 61.

Table 61 - CreateUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
List of Domains	M	M(=)			
List of Program Invocations	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.8.1.1 Argument

This parameter shall convey the parameters of the CreateUnitControl service request.

13.8.1.1.1 Unit Control Name

This parameter, of type Identifier, is the name that shall be assigned to the newly created Unit Control object.

13.8.1.1.2 List of Domains

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be referenced by the &Domains field of the Unit Control object.

13.8.1.1.3 List of Program Invocations

This parameter, of type list of Identifier, shall specify zero or more Program Invocations that are to be referenced by the &ProgramInvocations field of the Unit Control object.

13.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.8.2 Service procedure

The MMS server shall perform the following actions.

a) Create a Unit Control object and assign it the name as supplied in the Unit Control Name parameter.

b) For each element of the List of Domains parameter (if any), add a reference to that Domain to the &Domains field of the Unit Control object.

c) For each element of the List of Program Invocations parameter (if any), add a reference to that Program Invocation to the &ProgramInvocations field of the Unit Control object.

d) Return a Result(+).

13.9 AddToUnitControl service

The AddToUnitControl service is used by an MMS client to request an MMS server to add Domains and/or Program Invocations to the Unit Control object.

13.9.1 Structure

The structure of the component service primitives is shown in Table 62.

Table 62 - AddToUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
List of Domains	M	M(=)			
List of Program Invocations	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.9.1.1 Argument

This parameter shall convey the parameters of the AddToUnitControl service request.

13.9.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object in the VMD whose list of constituents is to be altered.

13.9.1.1.2 List of Domains

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be added to the &Domains field of the Unit Control object.

13.9.1.1.3 List of Program Invocations

This parameter, of type list of Identifier, shall specify zero or more Program Invocations that are to be added to &ProgramInvocations field of the Unit Control object.

13.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.9.2 Service procedure

The MMS server shall perform the following actions.

a) For each element of the List of Domains parameter (if any), add a reference to that Domain to the &Domains field of the Unit Control object.

b) For each element of the List of Program Invocations parameter (if any), add a reference to that Program Invocation to the &ProgramInvocations field of the Unit Control object.

c) Return a Result(+)

13.10 RemoveFromUnitControl service

The RemoveFromUnitControl service is used by an MMS client to request an MMS server to remove Domains, or Program Invocations, or both from the Unit Control object.

13.10.1 Structure

The structure of the component service primitives is shown in Table 63.

Table 63 - RemoveFromUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
List of Domains	M	M(=)			
List of Program Invocations	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.10.1.1 Argument

This parameter shall convey the parameters of the RemoveFromUnitControl service request.

13.10.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object whose list of constituents is to be altered.

13.10.1.1.2 List of Domains

This parameter, of type list of Identifier, shall specify zero or more Domains that are to be removed from the &Domains field of the Unit Control object.

13.10.1.1.3 List of Program Invocations

This parameter, of type list of Identifier, shall specify zero or more Program Invocations that are to be removed from the &ProgramInvocations field of the Unit Control object.

13.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.10.2 Service procedure

The MMS server shall perform the following actions.

a) For each element of the List of Domains parameter (if any), remove the reference to that Domain from the &Domains field of the Unit Control object.

b) For each element of the List of Program Invocations parameter (if any), remove the reference to that Program Invocation from the &ProgramInvocations field of the Unit Control object.

c) Return a Result(+)

13.11 GetUnitControlAttributes service

This service is used by an MMS client to request an MMS server to provide the list of constituent Domains and Program Invocations of a Unit Control object.

13.11.1 Structure

The structure of the component service primitives is shown in Table 64.

Table 64 - GetUnitControlAttributes service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
List of Domains			M	M(=)	
List of Program Invocations			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.11.1.1 Argument

This parameter shall convey the parameter of the GetUnitControlAttributes service request.

13.11.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object for which the attributes are to be obtained.

13.11.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

13.11.1.2.1 List of Domains

This parameter, of type list of Identifier, shall specify the names of the Domains that are referenced by the &Domains field of the Unit Control object.

13.11.1.2.2 List of Program Invocations

This parameter, of type list of Identifier, shall specify the names of the Program Invocations that are referenced by the &ProgramInvocations field of the Unit Control object.

13.11.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.11.2 Service procedure

The MMS server shall perform the following actions.

a) For each entry in the &ProgramInvocations field of the Unit Control object, verify that the Program Invocation exists. If the Program Invocation does not exist, remove its reference from the &ProgramInvocations field of the Unit Control object.

b) For each entry in the &Domains field of the Unit Control object, verify that the Domain exists. If the Domain does not exist, remove its reference from the &Domains field of the Unit Control object.

c) Return a Result(+) with the List of Domains parameter and List of Program Invocations parameter as specified in the &Domains field and &ProgramInvocations field of the Unit Control object.

NOTE Following the model of Unit Control object given in this part of ISO 9506, it is possible that the list of constituents of a Unit Control object may become inconsistent with the actual set of Domains and Program Invocations, e.g. following the explicit deletion of a Domain. The service procedure of this subclause is intended to reestablish consistency for this Unit Control object prior to completion of the service. A real implementation may choose to maintain consistency at all times by employing a set of inverse references in each Domain and Program Invocation. However, this is not required. An alternate implementation technique could be to implement the references within the Unit Control object by name, reestablishing consistency only when required.

13.12 LoadUnitControlFromFile service

The LoadUnitControlFromFile service is used by an MMS client to request an MMS server to create a Unit Control object and load the Unit Control object using information available locally or from a third party.

13.12.1 Structure

The Structure of the component service primitives is shown in Table 65.

Table 65 - LoadUnitControlFromFile service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
File Name	M	M(=)			
Third Party	U	U(=)			tpy
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Initiate Unit Control Error			M	M(=)	
Domain Name			S	S(=)	
Program Invocation Name			S	S(=)	

13.12.1.1 Argument

This parameter shall convey the parameters of the LoadUnitControlFromFile service request.

13.12.1.1.1 Unit Control Name

This parameter shall specify the name of the Unit Control object to be loaded.

13.12.1.1.2 File Name

This parameter, of type FileName, shall specify the name of the file containing the information to be loaded.

13.12.1.1.3 Third Party

This parameter, of type ApplicationReference, shall specify the application reference of the Application Process through which the named file may be accessed. Support of processing for this parameter is an implementation option that shall be implemented if support for the **tpy** parameter conformance building block is claimed. If it is implemented, its use is a user option. If this parameter is absent, the MMS server shall attempt to access the requested file directly.

13.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.12.1.3 Result (-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure. When failure is indicated, one of the following parameters shall be returned.

13.12.1.3.1 Domain Name

This parameter shall indicate the Domain that was being created when the error was detected. Either this parameter or the Program Invocation Name parameter may be present, but not both.

13.12.1.3.2 Program Invocation Name

This parameter shall indicate the Program Invocation that was being created when the error was detected. Either this parameter or the Domain Name parameter may be present but not both.

13.12.2 Service procedure

13.12.2.1 Preconditions

The MMS server shall verify that the Unit Control object of the specified name does not exist.

13.12.2.2 Actions

If a third party is specified, establish an association with that application if none exists; thereafter take appropriate action to cause the named Unit Control object to be loaded. If no third party is specified, perform the necessary steps to obtain the file through local means and load it into the specified Unit Control object. If the loading is successful, return a Result(+); otherwise return a Result(-) indicating in the Initiate Unit Control Error parameter the reason for failure.

13.13 StoreUnitControlToFile service

The StoreUnitControlToFile service is used by an MMS client to request an MMS server to store the Domains and Program Invocations of a Unit Control object either at a third party site or locally.

13.13.1 Structure

The structure of the component service primitives is shown in Table 66.

Table 66 - StoreUnitControlToFile service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
File Name	M	M(=)			
Third Party	U	U(=)			tpy
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

13.13.1.1 Argument

This parameter shall convey the parameters of the StoreUnitControlToFile service request.

13.13.1.1.1 Unit Control Name

This parameter shall specify the name of the Unit Control object for which the content is to be stored.

13.13.1.1.2 File Name

This parameter, of type FileName, shall specify the name of the file in which the information is to be stored.

13.13.1.1.3 Third Party

This optional parameter, of type ApplicationReference, shall specify the application reference of the Application Process on which the file store resides that is to receive the contents of the specified Unit Control object. Support of processing for this parameter is an implementation option that shall be implemented if support for the **tpy** parameter conformance building block is claimed. If it is implemented, its use is a user option. If this parameter is absent, the method of storing this file shall be a local matter.

13.13.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.13.1.3 Result (-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

13.13.2 Service procedure

13.13.2.1 Preconditions

The MMS server shall verify that the Unit Control object of the specified name exists.

If this condition is not satisfied, the MMS server shall return a Result(-).

13.13.2.2 Actions

If the third party parameter has been provided, the MMS server shall establish an association with that application if none exists; thereafter it shall take appropriate action to cause the named Unit Control object to be stored at the third party site. If no third party is specified, the MMS server shall perform the necessary steps to store the Unit Control object in the file specified through local means. If the process is successful, return a Result(+); otherwise return a Result(-).

13.14 DeleteUnitControl service

This service is used by an MMS client to request an MMS server to delete the Unit Control object and all its constituent elements.

13.14.1 Structure

The structure of the component service primitives is shown in Table 67.

Table 67 - DeleteUnitControl service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Unit Control Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Domain Name			S	S(=)	
Program Invocation Name			S	S(=)	

13.14.1.1 Argument

This parameter conveys the parameter of DeleteUnitControl service.

13.14.1.1.1 Unit Control Name

This parameter, of type Identifier, shall identify the Unit Control object that is to be deleted with its constituent elements.

13.14.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

13.14.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure. When failure is indicated, one of the following parameters shall be returned.

13.14.1.3.1 Domain Name

This parameter shall indicate the Domain whose deletion was being attempted when the error was detected. Either this parameter or the Program Invocation Name parameter shall be selected.

13.14.1.3.2 Program Invocation Name

This parameter shall indicate the Program Invocation whose deletion was being attempted when the error was detected. Either this parameter or the Domain Name parameter shall be selected.

13.14.2 Service procedure

The MMS server shall perform the following actions.

a) For each entry in the &ProgramInvocations field of the Unit Control object perform the following actions.

1) Verify that the Program Invocation exists. If the Program Invocation does not exist, remove its reference from the &ProgramInvocations field of the Unit Control object and skip the remainder of this step for this entry.

2) Verify that the Program Invocation is not in the **running** state. If this condition is not satisfied, return a Result(-) and skip the remainder of the procedure.2

3) Perform a DeleteProgramInvocation service procedure as specified in 12.3 and remove the reference to this Program Invocation from the &ProgramInvocations field of the Unit Control object. If this procedure fails, return a Result(-) and skip the remainder of the procedure.

b) For each entry in the &Domains field of the Unit Control object perform the following actions.

1) Verify that the Domain exists. If the Domain does not exist, remove its reference from the &Domains field of the Unit Control object and skip the remainder of this step for this entry.

2) Verify that the Domain is not in the **in-use** state. If this condition is not satisfied, return a Result(-) and skip the remainder of the procedure.

3) Perform a DeleteDomain service procedure as specified in 11.12 and remove the reference to this Domain from the &Domains field of the Unit Control object. If this procedure fails, return a Result(-) and skip the remainder of the procedure.

c) Delete the Unit Control object from the VMD.

d) Return a Result(+).

If the procedure returns a Result(-), the Unit Control object may have had some of its Domains and Program Invocations deleted. In this case, the Delete Unit Control Error parameter shall indicate the Domain or Program Invocation on which the procedure stopped, and the current contents of the Unit Control object shall indicate the Domains and Program Invocations that still remain on the Unit Control object.

NOTE Following the model of Unit Control object given in this part of ISO 9506, it is possible that the list of constituents of a Unit Control object may become inconsistent with the actual set of Domains and Program Invocations, e.g. following the explicit deletion of a Domain. The service procedure of this subclause is intended to reestablish consistency for this Unit Control object prior to completion of the service. A real implementation may choose to maintain consistency at all times by employing a set of inverse references in each Domain and Program Invocation. However, this is not required. An alternate implementation technique could be to implement the references within the Unit Control object by name, reestablishing consistency only when required.

14 Variable Access Services

This clause provides object models for the following objects:

UNNAMED-VARIABLE
NAMED-VARIABLE
NAMED-VARIABLE-LIST
NAMED-TYPE

This clause specifies the following services:

Read
Write
InformationReport
GetVariableAccessAttributes
DefineNamedVariable
DeleteVariableAccess
DefineNamedVariableList
GetNamedVariableListAttributes
DeleteNamedVariableList
DefineNamedType
GetNamedTypeAttributes
DeleteNamedType

The variable access services provide facilities that allow a MMS-client to access typed variables defined at the VMD. These facilities are provided by four objects in the VMD model, and by twelve services that operate upon these objects. Subclause 14.1 describes the MMS model for variable access. This is followed in 14.2 by a description of MMS types, in 14.3 by a description of MMS-definable alternate access, in 14.4 by a description of MMS data values, and in 14.5 by a description of parameters that specify access to variables. The variable access services are described in 14.6 through 14.17. Finally 14.18 describes static conformance requirements that apply for this clause.

CLARIFICATION: This clause describes the mapping between virtual objects, called MMS variables, and real objects, called "real" variables. MMS variables are not variables in the usual sense, but rather represent access paths to the underlying real objects. MMS variables do not have a true "value" attribute, but through the V-Put and V-Get functions they provide an access method to the value of the underlying real variables. These real objects may be true system variables, or they may be system constants, that is configuration parameters, or they may represent system procedures that generate a value, or other objects. Real variables may have addresses and may be described as contiguous or noncontiguous. Such concepts do not apply to MMS variables. In this context the word "real" means "having concrete existence", not "floating-point".

14.1 The MMS Variable Access Model

A "variable" is an abstract element of the VMD that is capable of providing (when read) or accepting (when written), or both, a typed data value. A "type description" is an abstract description of the class of data that may be conveyed by a variable's value. A variable's type description determines its abstract syntax, its range of possible values, and its representation while being communicated using MMS.

The MMS variable access services have as their focal point the reading and writing of values of variables (or parts of variables) residing at the VMD. The following paragraphs describe the MMS variable access model by describing the variable access objects and by describing the services that create, manipulate, or destroy these objects.

An MMS server that allows access to one or more of its real variables using the MMS variable access services shall provide a mapping between the real variables for which access is allowed and one or more MMS variable access objects.

NOTE Prior to reading the remainder of the variable access model, the reader may find it helpful to review "Guidance To Implementors" in 14.19.

14.1.1 Objects that Describe Variables

MMS defines two objects that describe the mapping between an MMS variable and a real variable at the VMD. They are the Unnamed Variable object and the Named Variable object. These objects provide for two distinct levels of abstraction.

The abstraction represented by the Unnamed Variable object is very close to the physical architecture of the real system. Unnamed Variable objects model device-dependent aspects of an addressable facility of the VMD. An Unnamed Variable object provides a mapping between the inherent type description represented by an address and the MMS representation for that type description. An Unnamed Variable object's existence corresponds with the existence of the VMD. This object is only available when the **vadr** parameter conformance building block is supported.

The abstraction represented by the Named Variable object models the application's view of a real variable at the VMD. A Named Variable object's existence may be specified to correspond with the life of a Domain, of an application association or of the VMD. The Named Variable object is only available when the **vnam** parameter conformance building block is supported.

Though both Named Variable and Unnamed Variable objects may be used to model real variables at the VMD, there are significant differences between these objects and the facilities that they provide for the MMS-client.

a) The Named Variable object describes access to the real variable using an application process determined name, whereas the Unnamed Variable object describes access to the real variable using a device-specific address.

b) The Named Variable object may be used to describe a variable that either does not have a known address (a computed variable) or that has an address that is not made public. The Unnamed Variable object requires a known, fixed address for the variable.

c) A Named Variable object may be used to describe a set of one or more application-related data elements that are accessed using a single name as a single operation (see 14.1.1.1), or to describe a more explicit type description for a set of one or more Unnamed Variable objects, or both. An Unnamed Variable object, on the other hand, models access to the built-in, implementation-dependent aspects of an addressable facility of the VMD.

This International Standard provides services that allow a MMS-client to define Named Variable objects in terms of Unnamed Variable objects. These services are primarily intended for use in bridging the gap between the two environments for devices that do not directly support the **vnam**-only (see 14.19.1) environment, whether due to age or simplicity, or both.

14.1.1.1 Requirements for Access to MMS Variables

The mapping of a Named Variable object or an Unnamed Variable object to a real variable shall ensure that access (to the real variable, using the object) either succeeds or fails; partial success shall not be reported.

NOTE In the case of write failure, for example due to a hardware fault in the real system, a partial result from the write may be visible to a subsequent access, either local or remote.

Additionally, the MMS server should guarantee, if possible, that access to an MMS variable is not interruptible. In other words, read access should return a value representing a single logical instant in the state of the VMD and write access should alter the state of the VMD at a single logical instant. Since such guarantees may not be possible, or if possible for some variables, may not be possible for all variables, the static conformance statement for the MMS server shall state whether uninterruptible access is supported and, if supported, under what constraints it is guaranteed.

14.1.1.2 Relationship Between MMS Variables and the Real System

The relationship between a real variable and the MMS object that is used to access it is specified by a pair of abstract functions. These functions are described below.

14.1.1.2.1 The V-Get Function

The V-Get function obtains the value of an MMS variable from the state of the VMD. (Here, state encompasses all aspects of the VMD's operation that relate to the mapping of the MMS variable to the real system.) The parameters of the V-Get function are the state of the VMD, the variable's access method (including its address if the access method is **public**), its type description and any alternate access that applies for the access. Failure of the conditions specified in the Access Control List object(s) for service class = READ requires the access of the MMS variable to fail.

If the access succeeds, the result is a value containing the accessed data elements of the real variable. The abstract type description of this value is specified by the type specification (14.2) and alternate access (14.3).

In the case of access failure, the result is a reason for failure. (See Data Access Error in 14.4.)

14.1.1.2.2 The V-Put Function

The V-Put function updates the state of the VMD with the value of an MMS variable. (Once again, state encompasses all aspects of the VMD's operation that relate to the mapping of the MMS variable to the real system). The parameters of the V-Put function are the variable's value, access method, type description and alternate access (if applicable). Given successful access, the MMS variable's type description and alternate access, if applicable, are applied to the MMS value to compute the real value of the variable and the result is used to update the state of the VMD. If this update fails, the resulting state of the VMD is unspecified and the access fails, giving a reason for failure. Failure of the conditions specified in the Access Control List object(s) for service class = WRITE requires the access of the MMS variable to fail.

14.1.1.3 Enforcement of Variable Access Privilege

Protection requirements (if any) for an MMS variable are inherited from the underlying real variable. These requirements are established by the MMS server, based on local criteria. They are not specified by this part of ISO 9506. These protection requirements are combined with the explicit protection requirements of the Access Control List object for service class = READ or WRITE as appropriate. The value OBJECT-ACCESS-DENIED for the Data Access Error shall be used to indicate that a request for variable access has been denied due to insufficient privilege.

14.1.2 The Unnamed Variable object

The Unnamed Variable object describes the mapping of a single unnamed MMS variable to a real variable existing at a known and fixed address within the VMD. This mapping shall be such that the requirements of 14.1.1.1 are satisfied.

An Unnamed Variable object is never created or destroyed. Its existence is inherent in the architecture of the VMD. When **vadr** is supported, the content of every public (remotely visible) address is modeled as an Unnamed Variable object.

Example: A general-purpose, byte-addressable device might assign the type "octet" to each public address. A special-purpose device with a small number of addresses might assign a specific type, based on the known use of the content of a specific address.

The attributes of the Unnamed Variable object are specified below, followed by a brief description of the services that operate on this object.

```
UNNAMED-VARIABLE ::= CLASS {
    &address            Address,
    &accessControl      ACCESS-CONTROL-LIST,
    &typeDescription    TypeDescription,
    &value              Data,
    &accessMethod       ENUMERATED {
        public              (0) }
-- The field '&accessMethod' shall have a value equal to public.
    }
```

14.1.2.1 &address

The &address field shall provide the location of the real variable in the system that supports the VMD.

```
Address ::= CHOICE {
    numericAddress      [0] IMPLICIT Unsigned32,
    symbolicAddress     [1] MMSString,
    unconstrainedAddress [2] IMPLICIT OCTET STRING }
```

14.1.2.2 &accessControl

The &accessControl field shall specify an Access Control List object that provides conditions under which this Unnamed Variable may be read or written.

14.1.2.3 &typeDescription

The &typeDescription field of an Unnamed Variable shall indicate one of the choices of the TypeDescription type. The TypeDescription type is described in 14.2.2.

14.1.2.4 &value

The &value field of an Unnamed Variable shall indicate the value associated with this variable. The choice of the Data type to be used shall be the same choice as is made for the &typeDescription field.

14.1.2.5 &accessMethod

The &accessMethod field for an Unnamed Variable object shall specify **public** access.

14.1.2.6 Operations On The Unnamed Variable object

The services that operate on the Unnamed Variable object are listed below.

a) Read - This service uses the V-Get function to obtain the current &value of a real variable described by the Unnamed Variable object;

b) Write - This service uses the V-Put function to replace the current &value of a real variable described by the Unnamed Variable object;

c) InformationReport - This service uses the V-Get function to obtain the current &value of a real variable described by the Unnamed Variable object;

d) GetVariableAccessAttributes - This service returns the attributes of a Unnamed Variable object.

14.1.3 The Named Variable object

The Named Variable object describes the mapping between an MMS variable and a real, application-defined, variable at the VMD. This mapping shall be such that the requirements of 14.1.1.1 are satisfied.

NOTE It is recommended that a real variable have one and only one mapping to a Named Variable object. This International Standard does not require this.

The attributes of the Named Variable object are specified below, followed by a brief description of the services that operate on this object.

```
NAMED-VARIABLE ::= CLASS {
    &name                ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
    &accessControl       ACCESS-CONTROL-LIST,
    &typeDescription     TypeDescription,
    &value               Data,
    &accessMethod        ENUMERATED {
        public               (0),
        anythingElse         (1),
        ...    },
    &address             Address OPTIONAL,
-- The presence of this field shall correspond to the
-- field &access Method having a value equal to public.
-- The absence of this field shall correspond to the
-- field &accessMethod having a value equal to anything except public.
--
-- The following field shall occur
--    if and only if the sem CBB has been negotiated.
IF (sem)
    &meaning             ObjectName OPTIONAL
ENDIF
    }
```

14.1.3.1 &name

The &name field uniquely identifies a Named Variable object. A &name is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope.

14.1.3.2 &accessControl

The &accessControl field shall specify an Access Control List object that provides conditions under which this Named Variable may be read, written, deleted, or have its access control changed.

14.1.3.3 &typeDescription

The &typeDescription field of a Named Variable object shall indicate one of the choices of the TypeDescription type. The TypeDescription type is described in 14.2.2.

14.1.3.4 &value

The &value field of a Named Variable shall indicate the value associated with this variable. The choice of the Data type to be used shall be the same choice as is made for the &typeDescription field.

14.1.3.5 &accessMethod

The &accessMethod field for an Named Variable object shall specify the mode of access. If the Address is declarable (and obtainable) using MMS services, the &accessMethod field shall have the value **public**, and the Address attribute shall be defined and available to MMS clients requesting the attributes of the Named Variable object. Otherwise, the value of this field is a local issue. The **public** access method shall not be available unless **vadr** is supported.

An MMS server may declare an MMS variable that exists only at the instant of access. Such a variable does not have an address per se. It shall be defined by the MMS server using local means and shall have an &accessMethod other than **public**. Also, an MMS server may declare an MMS variable that does have an address, but choose not to reveal this address to MMS clients. Such a variable shall also be locally defined with an &accessMethod other than **public**.

If the &accessMethod is **public** the following field shall appear. If the &accessMethod is anything but **public**, the following field shall not appear.

14.1.3.6 &address

The &address field shall provide the location of the real variable in the system that supports the VMD. This field shall be present if and only if the value of the &accessMethod is **public**.

The data elements of the variable, as described by the &typeDescription field, are located in contiguous addresses, starting with the first data element at the address specified by this field. The types of the data elements shall be compatible with the types of the data elements of the Unnamed Variable objects that are encompassed by these contiguous addresses. The determination of type "compatibility" is a local issue.

NOTE 1 The intent is to allow the MMS server to refuse to accept a definition containing a type conflict, for example a request to define a known boolean as a floating-point. The definition is left open in order to allow the MMS server to accept a definition requesting a tightening of type specification, for example a declaration that a "word" (bitstring) contains an integer.

NOTE 2 This International Standard does not provide for allocation of real variables in the VMD. MMS services may be used to describe where and how a real variable has been allocated. However, such usage requires detailed knowledge of the specific VMD implementation, including the specific application of the VMD in which the real variable is declared.

NOTE 3 The requirement of contiguity of address is only for variables whose &accessMethod is **public**.

14.1.3.7 &meaning

This field may be used to store the name of a Named Type object; the name of this object may convey semantics about the Named Variable object. The value of this field is set by the DefineNamedVariable service and may be reported by the GetVariableAccessAttributes service if the **sem** CBB has been negotiated. This field may also be part of a predefined Named Variable.

14.1.3.8 Operations On The Named Variable object

The services that operate upon the Named Variable object are listed below.

a) Read - This service uses the V-Get function to obtain the &value of a real variable described by the Named Variable object;

b) Write - This service uses the V-Put function to replace the &value of a real variable described by the Named Variable object;

c) InformationReport - This service uses the V-Get function to obtain the &value of a real variable described by the Named Variable object;

d) DefineNamedVariable - This service creates a Named Variable object;

e) GetVariableAccessAttributes - This service returns the attributes of a Named Variable object;

f) DeleteVariableAccess - This service deletes a Named Variable object.

14.1.4 The Named Variable List object

The Named Variable List object describes access to multiple MMS variables using a single name.

The Named Variable List object shall specify a mapping of a single MMS name to a list of independent MMS variables. The Named Variable List object is designed to maintain client awareness of the independence of the underlying real variables. This object provides a mechanism to avoid repeated transfer of frequently used variable access lists. Access using a Named Variable List succeeds or fails at the level of the members of the list. Therefore partial success may be reported.

Each element of this list shall be either a Named Variable object or an Unnamed Variable object. Access to the individual elements of the list is regulated by the requirements of 14.1.1.1. Access to the list as a whole is not governed by this requirement. Thus, access using the Named Variable List object shall report success or failure for each object referenced by the list. Access using a Named Variable List object is analogous to independent accesses using the list's referenced variable access objects.

The attributes of the Named Variable List object are specified below, followed by a brief description of the services that operate on this object.

```
NAMED-VARIABLE-LIST ::= CLASS {
    &name                ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
    &accessControl       ACCESS-CONTROL-LIST,
    &listOfVariables     VARIABLE-LIST-ITEM }
```

14.1.4.1 &name

The &name field uniquely identifies a Named Variable List object. A &name is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope.

14.1.4.2 &accessControl

The &accessControl field shall specify an Access Control List object that provides conditions under which this Named Variable List object may be read, written, deleted, or have its access control changed.

14.1.4.3 &listOfVariables

The &listOfVariables field identifies one or more VARIABLE-LIST-ITEM objects. Each of these items identifies a variable (named or unnamed) and an optional alternate access specification.

```
VARIABLE-LIST-ITEM ::= CLASS {
-- one and only one of the following two lines shall appear
IF ( vadr )
    &unnamedItem         UNNAMED-VARIABLE OPTIONAL,
ENDIF
IF ( vnam )
    &namedItem           NAMED-VARIABLE OPTIONAL,
ENDIF
IF (valt)
    -- the following specification may be included
    &alternateAccess     AlternateAccess OPTIONAL
ENDIF
    }
```


14.1.4.3.1 &unnamedItem

The &unnamedItem field provides a specification for access to a variable identified by an Unnamed Variable object. This choice may be present only if the **vadr** parameter CBB has been negotiated.

14.1.4.3.2 &namedItem

The &namedItem field provides a specification for access to a variable identified by a Named Variable object. This choice may be present only if the **vnam** parameter CBB has been negotiated.

14.1.4.3.3 &alternateAccess

The &alternateAccess field provides an Alternate Access Specification to the (Named or Unnamed) Variable object. This field may be present only if the **valt** parameter CBB has been negotiated.

14.1.4.4 Operations On The Named Variable List object

The services that operate upon the Named Variable List object are listed below.

a) Read - This service uses the &listOfVariable attribute in order to determine the variables that should be read and performs the Read service on these variables;

b) Write - This service uses the &listOfVariable attribute in order to determine the variables that are to be written and performs the Write service on these variables;

c) InformationReport - This service uses the &listOfVariable attribute in order to determine the variables to be reported and performs an Information Report service on these variables;

d) DefineNamedVariableList - This service creates a Named Variable List object;

e) GetNamedVariableListAttributes - This service returns the attributes of a Named Variable List object;

f) DeleteNamedVariableList - This service deletes a Named Variable List object.

14.1.5 The Named Type object

The Named Type object provides for the assignment of a name to an MMS type description. This object is available only when both the **vadr** and the **vnam** parameter conformance building blocks are supported.

The attributes of the Named Type object are specified below, followed by a brief description of the services that operate on this object.

```
NAMED-TYPE ::= CLASS {
      &name                ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
      &accessControl       ACCESS-CONTROL-LIST,
      &typeDescription     TypeDescription,
--
-- The following field shall occur
--    if and only if the sem CBB has been negotiated.
IF (sem)
      &meaning             ObjectName OPTIONAL
ENDIF
      }
```

14.1.5.1 &name

The &name field uniquely identifies a Named Type object. A &name is an MMS Object Name and may be defined with VMD-specific, Domain-specific, or Application Association-specific scope.

14.1.5.2 &accessControl

The &accessControl field shall specify an Access Control List object that provides conditions under which this Named Type may be deleted or have its access control changed.

14.1.5.3 &typeDescription

The &typeDescription field shall indicate a choice from the TypeDescription abstract type. The TypeDescription type is described in 14.2.2.

14.1.5.4 &meaning

This field may be used to store the name of another Named Type object; the name of this object may convey semantics about this Named Type object. The value of this field is set by the DefineNamedType service and may be reported by the GetNamedTypeAttributes service if the **sem** CBB has been negotiated. This field may also be part of a predefined Named Type.

14.1.5.5 Operations On The Named Type object

The services that operate upon the Named Type object are listed below.

a) DefineNamedType - This service creates a Named Type object;

b) GetNamedTypeAttributes - This service returns the attributes of a Named Type object;

c) DeleteNamedType - This service deletes a Named Type object;

d) Read, Write, DefineNamedVariable, DefineNamedVariableList, DefineNamedType - These services use the &typeDescription field in order to resolve a Type Name parameter that may be included in the service request.

14.2 Specification of Types

All MMS variables are typed. A variable's type description, as contained in the &typeDescription field of the Named Variable or Unnamed Variable object, or as specified by the Variable Description parameter's Type Specification parameter at the time of access, shall provide a specification of the abstract syntax and range of possible values of the variable, and also provide the basis upon which alternate access to the variable may be specified.

The type description of a variable may be simple, specifying access to a single data element, or complex, specifying access to a group of related types. For each complex variable (array or structure), from the information available in a variable's access method (and address, if applicable), and its type description, the MMS server shall be able to locate every simple data element of the real variable.

14.2.1 TypeDescription type

The &typeDescription field of the Unnamed Variable object, the Named Variable object, or the Named Type object shall indicate one of the choices of the TypeDescription type.

```
 TypeDescription ::= CHOICE {
IF ( str1 )
     array                  [1] IMPLICIT SEQUENCE {
          packed                 [0] IMPLICIT BOOLEAN DEFAULT FALSE,
          numberOfElements       [1] IMPLICIT Unsigned32,
          elementType            [2] TypeSpecification },
ENDIF
```



```
IF ( str2 )
    structure           [2] IMPLICIT SEQUENCE {
        packed              [0] IMPLICIT BOOLEAN DEFAULT FALSE,
        components          [1] IMPLICIT SEQUENCE OF SEQUENCE {
            componentName       [0] IMPLICIT Identifier OPTIONAL,
            componentType       [1] TypeSpecification } },
ENDIF
-- Simple Size Class
    boolean             [3] IMPLICIT NULL,          -- BOOLEAN
    bit-string          [4] IMPLICIT Integer32,     -- BIT-STRING
    integer             [5] IMPLICIT Unsigned8,     -- INTEGER
    unsigned            [6] IMPLICIT Unsigned8,     -- UNSIGNED
    floating-point      [7] IMPLICIT SEQUENCE {
        format-width        Unsigned8,                  -- number of bits of
                                                        -- floating point value
                                                        -- including sign, exponent,
                                                        -- and fraction
        exponent-width      Unsigned8                   -- size of exponent in bits
        },
                            -- [8] is reserved
    octet-string        [9] IMPLICIT Integer32,     -- OCTET-STRING
    visible-string      [10] IMPLICIT Integer32,    -- VISIBLE-STRING
    generalized-time    [11] IMPLICIT NULL,         -- GENERALIZED-TIME
    binary-time         [12] IMPLICIT BOOLEAN,      -- BINARY-TIME
    bcd                 [13] IMPLICIT Unsigned8,    a -- BCD
    objId               [15] IMPLICIT NULL,
    ...,
    mMSString           [16] Integer32              -- MMS String
    }
```

14.2.2 Type Description parameter

The structure of The Type Description parameter is given in Table 68.

Table 68 - Type Description parameter

Parameter Name	Req/Rsp	Ind/Cnf	CBB
Kind of Type	M	M(=)	
Array	S	S(=)	str1
Packed	M	M(=)	vadr
Number of Elements	M	M(=)	
Type Specification	M	M(=)	str2
Structure	S	S(=)	vadr
Packed	M	M(=)	
List of Components	M	M(=)	valt
Component Name	M	M(=)	
Type Specification	M	M(=)	
Simple	S	S(=)	
Class	M	M(=)	
Size	C	C(=)	

A type is described by a recursively specified parameter, the Type Specification. The Type Description parameter describes a branching tree, called a type tree. The leaves of this tree are the simple data elements of the variable described by the type. If

a type describes a complex variable (an array or a structure), the type tree will have one or more non-leaf nodes, each of which represents a complex type constructed from the (possibly complex) types represented by the subordinate nodes of the type tree.

The parameters of the Type Description parameter are as follows.

14.2.2.1 Kind Of Type

The Kind Of Type parameter shall indicate the selection chosen to describe this node of the type tree. The values for this parameter are:

ARRAY - shall indicate that the Array parameter is selected.

STRUCTURE - shall indicate that the Structure parameter is selected.

SIMPLE - shall indicate that the Simple parameter is selected.

14.2.2.2 Array

This selection for the Type Description parameter shall indicate that the node being described is a complex type that is constructed from an ordered sequence of elements of a single type, with elements numbered from zero, the first element, and increasing.

NOTE From a modelling perspective, an array is described by the number of sub-trees specified by the Number Of Elements parameter, each having the type specified by the Type Specification parameter, and packing, as specified by the Packed parameter, and each immediately subordinate to the node of the type tree that specifies the array type. A given array element is identified by the position of its sub-tree under the array's type tree. The first array element is identified by zero, the second by one, and so forth, with the last element identified by the value of Number Of Elements minus one.

14.2.2.2.1 Packed

This parameter shall specify whether or not storage optimization rules are in effect for locating the data elements of this array (and its subordinate types). When false, unless defined subordinate to a type for which this attribute is true, storage optimization rules are not in effect. When true, storage optimization rules are in effect for the entire sub-tree described by the array type.

For Type Description parameters that specify the type of a Named Variable object having Access Method equal to **public** or that specify the type associated with an Unnamed Variable object, this parameter may be true or false, as applicable to the variable. Otherwise, this parameter shall be false.

NOTE The specific rules for locating the data elements of a storage optimized array are locally determined. These rules should ensure the integrity of surrounding data elements in the case of partial access, if allowed.

14.2.2.2.2 Number Of Elements

This parameter shall specify the number of elements of the array.

14.2.2.2.3 Type Specification

This parameter shall specify the type description of the array elements through a recursive reference to the Type Specification parameter.

14.2.2.3 Structure

This selection for the Type Description parameter shall specify that this node of the type tree describes a complex type that is constructed from an ordered list of one or more components, each of which may have a distinct type.

14.2.2.3.1 Packed

This parameter shall specify whether or not storage optimization rules are in effect for locating the data elements of this structure (and its subordinate types) or not. When false, unless defined subordinate to a type for which this attribute is true, storage optimization rules are not in effect. When true, storage optimization rules are in effect for the entire sub-tree described by the structure type.

For Type Description parameters that specify the type of a Named Variable object having Access Method equal to **public** or that specify the type associated with an Unnamed Variable object, this parameter may be true or false, as applicable to the variable. Otherwise, this parameter shall be false.

NOTE The specific rules for locating the data elements of a storage optimized structure are locally determined. These rules should ensure the integrity of surrounding data elements in the case of alternate access, if allowed.

14.2.2.3.2 List Of Components

The List Of Components parameter shall describe the components of the structure. At least one component shall be described.

14.2.2.3.2.1 Component Name

The Component Name parameter, of type Identifier, shall uniquely identify the component within the scope of the structure (node) to which the component is an immediate subordinate. This parameter is required if this node may be referenced by an alternate access specification. Otherwise, this parameter may be omitted.

14.2.2.3.2.2 Type Description

This parameter shall specify the type description of the structure component through a recursive reference to the Type Specification parameter.

14.2.2.4 Simple

This selection for the Type Description parameter shall indicate that a leaf node of the type tree is being described. Such a node shall contain the class of the data element represented by the node and, when applicable, the size (or precision) associated with the specific instance of the class. The class and size together serve to specify the range of possible values for the variable.

14.2.2.4.1 Class

The Class parameter of the Simple selection shall specify the class of the data element represented by the leaf node. The value of this parameter shall be chosen from one of the following values.

a) BOOLEAN - The definition of this MMS type is as specified for the boolean type in ISO/IEC 8824-1. The Size parameter shall be omitted.

b) BIT STRING - The definition of this type is as specified for the bitstring type in ISO/IEC 8824-1. The Size parameter shall specify the number of bits in the bit string and an indication of whether this is an absolute number (indicating a fixed-length bitstring) or a maximum number (indicating a variable-length bitstring).

c) INTEGER - The definition of this type is as specified for the integer type in ISO/IEC 8824-1. The Size parameter shall specify the number of bits (assuming twos-complement representation) required in order to allow representation of all possible distinguished values.

d) UNSIGNED - The definition of this type is as specified for the integer type in ISO/IEC 8824-1, with the exclusion of the negative whole numbers. The Size parameter shall contain the number of bits (assuming binary representation) required in order to allow representation of all possible distinguished values.

e) FLOATING POINT - This class defines a simple type with distinguished values that are the positive and negative real numbers, including zero, and including a representation for positive and negative infinity, and NaN (not a number). The

Size parameter shall specify in bits the format width, F, and the exponent width, E. The format width is the number of bits used to represent the floating point value including sign, exponent, and fraction.

NOTE 1 The terms "positive infinity", "negative infinity", "NaN", "format width" and "exponent width" are defined in the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Standard 754 - 1985). MMS allows any number of bits in the format and exponent. This includes, but is not limited to, the two basic formats defined in that standard. Additional information may be found in ISO 9506-2.

f) Reserved

NOTE 2 ISO/IEC 9506:1990 defined an additional choice, the REAL representation as defined in ISO/IEC 8824-1. That choice has been deprecated.

g) OCTET STRING - The definition of this type is as specified for the octetstring type in ISO/IEC 8824-1. The Size parameter shall contain the number of octets in the octetstring and an indication of whether this is an absolute number (indicating a fixed-length string) or a maximum number (indicating a variable-length string).

h) VISIBLE STRING - The definition of this type is as specified for the VisibleString type in ISO/IEC 8824-1. The Size parameter shall contain the number of characters in the string and an indication of whether this is an absolute number (indicating a fixed-length string) or a maximum number (indicating a variable-length string).

i) GENERALIZED TIME - The definition of this type is as specified for the GeneralizedTime type in ISO/IEC 8824-1. The Size parameter shall be omitted.

j) BINARY TIME - The definition of this type is as specified for the TimeOfDay type in this part of ISO 9506. The Size parameter shall indicate whether (true) or not (false) the date is to be included in values of the type.

k) BCD - This type shall consist of the set of distinguished values that are sequences of one or more numeric digits (0, 1, ... 9) from some character set. The Size parameter shall indicate the absolute number of digits of the type.

l) OBJECT IDENTIFIER - The definition of this type is as specified for the object identifier type in ISO/IEC 8824-1. The Size parameter shall be omitted.

m) MMS STRING - This type allows for a choice between an English character set, a European character set, or a general representation of characters from a specified set. The Size parameter shall contain the number of characters in the string and an indication of whether this is an absolute number (indicating a fixed-length string) or a maximum number (indicating a variable-length string).

14.2.2.4.2 Size

The presence (or not) of the Size parameter, as well as its meaning, is dependent upon the value of the Class parameter, as specified above. If Class specifies a variable length string type, Size shall be a negative integer determined by subtracting the maximum length of the string from zero.

14.2.3 Type Specification Parameter

The structure of the Type Specification parameter is given in Table 69.

Table 69 - Type Specification parameter

Parameter Name	Req/Rsp	Ind/Cnf	CBB
Kind of Type Type Name Type Description	M S S	M(=) S(=) S(=)	vnam

In the various variable access services, the type description of a variable or the definition of a Named Type shall be specified by the Type Specification parameter. This parameter itself describes a procedure by which a &typeDescription field is generated. This generation shall always be performed at the time of generation of the MMS variable object with the effect that all references to Named Type objects are resolved at the time of definition, and Type Descriptions never explicitly depend on Named Type objects.

NOTE As a consequence of this immediate evaluation rule, it is possible to define a VMD-specific variable in terms of a AA-specific Named Type. For example, a variable will contain a correct Type Description attribute even if a referenced AA-specific Named Type object is deleted because the application association disappears.

The parameters of the Type Specification parameter are as follows.

14.2.3.1 Kind Of Type

The Kind Of Type parameter shall indicate the selection that has been chosen to describe this node of the type tree. The values for this parameter are:

TYPE-NAME - shall indicate that the Type Name parameter is selected. This value shall not occur in a response or confirm service primitive, or in an InformationReport.indication primitive.

TYPE-DESCRIPTION - shall indicate that the TypeDescription parameter is selected.

14.2.3.2 Type Name

The Type Name selection for the Type Specification parameter shall indicate that this node of the type tree is to inherit its definition from the &typeDescription field of the Named Type object having a &name field equal to this parameter. (The Type Name is replaced by the value of the Named Type object's &typeDescription attribute).

14.2.3.3 Type Description

The Type Description selection for the Type Specification parameter shall indicate that this node of the type tree is to use the explicit Type Description attribute identified.

14.3 Specification of Alternate Access

As described in 14.2, all MMS variables are typed. A variable's type describes the abstract syntax and range of possible values of the real variable in the VMD. An alternate access description specifies an alternate view of this type. It may be used to alter the perceived abstract syntax of the variable (as seen using MMS services) or to restrict access to a subset of the range of possible values of the variable (partial access) or both.

Alternate access to a variable, as provided by the &accessMethod parameter of the Named Variable List object or as provided by the Alternate Access parameter of a specific access request, provides a mapping from (to) the view provided by the referenced MMS object to (from) the view that is desired by the access. This results in an indirect mapping to the real variable.

In the various variable access services, alternate access is represented by the presence of the Alternate Access parameter. It describes the derivation of the "derived type" that results from applying an alternate access to a type, whether derived or explicitly specified by a Type Specification. This derived type determines the abstract syntax of the Data parameter (14.4), when used to convey values associated with the alternate access.

The description of the Alternate Access parameter is based upon its relationship to an MMS variable's type.

14.3.1 Alternate Access parameter

The structure of the Alternate Access parameter is given in Table 70.

Table 70 - Alternate Access parameter

Conformance: valt Parameter Name	Req/Rsp	Ind/Cnf	CBB
List of Alternate Access Specification	M	M(=)	
Component Name	U	U(=)	
Kind of Selection	M	M(=)	
Select Alternate Access	S	S(=)	
Access Selection	M	M(=)	
Component	S	S(=)	str2
Index	S	S(=)	str1
Index Range	S	S(=)	str1
Low Index	M	M(=)	
Number of Elements	M	M(=)	
Alternate Access	M	M(=)	
Select Access	S	S(=)	
Access Selection	M	M(=)	
Component	S	S(=)	str2
Index	S	S(=)	str1
Index Range	S	S(=)	str1
Low Index	M	M(=)	
Number of Elements	M	M(=)	

14.3.1.1 List Of Alternate Access Selection

This parameter shall specify a list containing one or more Alternate Access Selection parameters. Each Alternate Access Selection parameter selects a node or, in the case of an array, a range of nodes, at the next higher nesting level of the type tree. This selection may be for the purpose of additional alternate access specification or for specifying access to the data elements represented by the selected nodes.

If the List Of Alternate Access Selection parameter contains more than one element, the derived type resulting from this parameter is a structure. The components of this derived type have component names and types as determined by the selections specified in the list.

If the List Of Alternate Access Selection parameter contains exactly one element, the derived type resulting from this parameter is determined by the selection specified for it, and the Component Name parameter shall not be specified.

14.3.1.1.1 Component Name

This parameter shall not be present if the List Of Alternate Access Selection specifies a single selection. Otherwise it is optional, and if present, shall specify the name of this component of the alternate access for use in specifying alternate access to the type derived from this application of the Alternate Access parameter.

14.3.1.1.2 Kind Of Selection

This parameter shall indicate whether access or further recursion in the alternate access specification is being specified. The possible values are:

SELECT-ALTERNATE-ACCESS - Indicates that further recursion in the alternate access specification is being specified and that the Select Alternate Access parameter is present.

SELECT-ACCESS - Indicates that access (read or write) is being specified and that the Select Access parameter is present.

14.3.1.1.3 Select Alternate Access

This parameter is selected for the Alternate Access Selection if one or more sub-trees at the next higher nesting level of the type tree are to be selected for further (recursive) alternate access specification. If the current node is an array, it shall select a single array element, a sub-range of array elements, or all array elements for further alternate access specification. If the current node is a structure, it shall select a single component of the structure for further alternate access specification. The parameters of Select Alternate Access are as follows.

14.3.1.1.3.1 Access Selection

This parameter shall indicate which of the selections for specifying alternate access has been taken. The possible values are:

COMPONENT - selects a single component of the structure as identified by the Component parameter, for alternate access.

INDEX - selects a single array element, as specified by the Index parameter, for alternate access.

INDEX-RANGE - selects an array of elements, as specified by the Index Range parameter, for alternate access.

14.3.1.1.3.2 Component

The Component parameter, of type Identifier, shall be selected if the current node of the type tree specifies a structure and the Access Selection parameter indicates COMPONENT. This parameter selects, for further alternate access specification, the specific structure component having Component Name equal to the Component parameter. The selected component shall be an array or a structure. The type tree node representing the selected structure component's type description is selected and the Alternate Access parameter is applied to that node. The derived type that results from applying the Component selection is determined by application of the Alternate Access parameter to the selected type tree node.

14.3.1.1.3.3 Index

The Index parameter, of type integer, shall be selected if the current node of the type tree specifies an array and the Access Selection indicates INDEX. Otherwise this parameter shall not be selected. It selects the specific array element for further alternate access specification. The selected component shall be an array or a structure. The type tree node representing the selected array element's type description is selected and the Alternate Access parameter is applied to this node. The derived type that results from applying the Index selection is determined by application of the Alternate Access parameter to the selected node.

14.3.1.1.3.4 Index Range

The Index Range parameter shall be selected if the current node of the type tree specifies an array and the Access Selection parameter indicates INDEX-RANGE. It selects a range of array elements (having array or structure type) for further alternate access specification. For each element of the selected range, in order of increasing index, the type tree node representing that element's type description is selected and the Alternate Access parameter is applied to that node. The derived type that results from

applying the Index Range selection is an array having elements of the type determined by application of the Alternate Access parameter to each of the selected nodes. (These elements are numbered from zero as for any other array).

14.3.1.1.3.4.1 Low Index

This integer parameter shall indicate the start of the index range. It shall be a valid index of the array. The specified element shall be the first element of the resulting derived array type and shall be numbered zero in that type.

14.3.1.1.3.4.2 Number Of Elements

This integer parameter shall indicate the number of elements to be included in the index range, including the element selected by Low Index. If this parameter has the value zero, all elements having index greater than or equal to Low Index and less than or equal to the maximum index for the array are selected. If greater than zero, all elements having index between Low Index and Low Index plus Number Of Elements minus one, inclusive are selected. Each of the selected elements shall be defined.

14.3.1.1.3.5 Alternate Access

The Alternate Access parameter shall specify additional alternate access at the selected node (or nodes).

14.3.1.1.4 Select Access

This parameter is selected for the Alternate Access Selection if one or more sub-trees at the next higher nesting level of the type tree are desired for access (read or write). If the current node is an array, it shall select a single array element or a sub-range of array elements for access. If the current node is a structure, it shall select a single component of the structure for access. The entire sub-tree (including all of its data elements) specified by the selected node (or nodes) is accessed. The parameters of Select Access are as follows.

14.3.1.1.4.1 Access Selection

This parameter shall indicate which of the selections for specifying access has been taken. The possible values are:

COMPONENT - selects a single component of the structure as identified by the Component parameter, for access.

INDEX - selects a single array element, as specified by the Index parameter, for access.

INDEX-RANGE - selects an array of elements, as specified by the Index Range parameter, for access.

14.3.1.1.4.2 Component

The Component parameter shall be selected if the current node of the type tree specifies a structure and the Access Selection parameter indicates COMPONENT. Otherwise, this parameter shall not be selected. It selects a specific structure component for access. The derived type resulting from the selection is the type of the selected structure component.

14.3.1.1.4.3 Index

The Index parameter shall be selected if the current node of the type tree specifies an array and the Access Selection parameter indicates INDEX. Otherwise, this parameter shall not be selected. It selects the specific array element to be accessed. The derived type resulting from the selection is the type of the selected array element.

14.3.1.1.4.4 Index Range

The Index Range parameter shall be selected if the current node of the type tree specifies an array and the Access Selection parameter indicates INDEX-RANGE. Otherwise, this parameter shall not be selected. It selects a range of array elements that are to be accessed. The derived type resulting from the Index Range selection is an array containing the selected elements, each having the type of the array element. (These elements are numbered from zero as for any other array).

14.3.1.1.4.4.1 Low Index

This integer parameter shall indicate the start of the index range. It shall be a valid index of the array. The specified element shall be the first element of the resulting derived array type and shall be numbered zero in that type.

14.3.1.1.4.4.2 Number Of Elements

This integer parameter shall indicate the number of elements to be included in the index range, including the element selected by Low Index. If this parameter has the value zero, all elements having index greater than or equal to Low Index and less than or equal to the maximum index for the array are selected. If greater than zero, all elements having index between Low Index and Low Index plus Number Of Elements minus one, inclusive are selected. Each of the selected elements shall be defined.

14.4 Specification of Data Values

The MMS Read, Write and InformationReport services are used to convey the desired (Write) or current (Read and InformationReport) value of an MMS variable or a set of MMS variables referenced by a Named Variable List object.

For Read and InformationReport, the current value of an MMS variable is obtained by applying the V-Get function (see 14.1.1.2), and the result is represented by the Access Result parameter.

For Write, the desired value of a variable is represented by the Data parameter. For an MMS variable, this value updates the current state of the VMD by application of the V-Put function (see 14.1.1.2). The result of this update, if successful, is represented by a simple confirmation. If the update fails, the result is represented by the Data Access Error parameter.

The Access Result, Data and Data Access Error parameters are specified below.

14.4.1 Access Result parameter

The Access Result parameter is used by the Read and InformationReport services in order to inform the client of the result of reading an MMS variable object.

The structure of this parameter is given in Table 71.

Table 71 - Access Result parameter

Parameter Name	Req/Rsp	Ind/Cnf	CBB
Success Data Access Error Data	M S S	M(=) S(=) S(=)	

14.4.1.1 Success

This parameter shall indicate whether (true) or not (false) the access succeeded, and shall specify the selection for the following parameters.

14.4.1.2 Data Access Error

If the Success parameter indicates that the access failed, this parameter shall contain the reason for failure. This parameter is described in 14.4.3.

14.4.1.3 Data

If the Success parameter indicates that the access succeeded, this parameter shall contain the value of the variable (or constructed variable). The abstract syntax of a Data value shall be determined by the derived type specified by the alternate access or by the variable's defined type (Named or Unnamed Variable object).

The parameters of the Data parameter are described in 14.4.2.

14.4.2 Data parameter

The Data parameter is used by the Read, Write and InformationReport services to convey the value of a variable.

The structure of the component parameters is shown in Table 72.

Table 72 - Data parameter

Parameter Name	Req/Rsp	Ind/Cnf	CBB
Kind of Data	M	M(=)	
Array	S	S(=)	str1
List of Data	M	M(=)	
Structure	S	S(=)	str2
List of Data	M	M(=)	
Simple	S	S(=)	
Class	M	M(=)	
Value	M	M(=)	

This parameter is recursively defined. It shall specify a branching tree. Each node of this tree corresponds to a node of the variable's type tree, or derived type tree after applying alternate access, see 14.2 and 14.3.

14.4.2.1 Kind Of Data

The value of the Kind Of Data parameter shall be ARRAY, STRUCTURE or SIMPLE depending upon whether the value at the current node of the type tree is an array, a structure, or a simple data element, respectively.

14.4.2.2 Array

The Array parameter shall be selected if Kind Of Data indicates an array. It shall specify an ordered list of Data parameters. Each element of this list provides the value of the corresponding element of the array. The elements of the list are ordered from array element zero and to the last array element.

14.4.2.3 Structure

The Structure parameter shall specify an ordered list of Data parameters. Each element of this list shall specify the value of the corresponding component of the structure. The elements of the list are ordered from the first component to the last component of the structure.

14.4.2.4 Simple

The Simple parameter shall be selected when Kind Of Data indicates a simple data element. It shall specify the class and value of a simple data element of the variable.

14.4.2.4.1 Class

This parameter shall indicate the class of data conveyed by the current value. Its possible values are BOOLEAN, BIT STRING, INTEGER, UNSIGNED, FLOATING POINT, OCTET STRING, VISIBLE STRING, GENERALIZED TIME, BINARY TIME, BCD, OBJECT IDENTIFIER, and MMS STRING as specified for the Simple type of the Type Description parameter, in 14.2.2.4.1.

14.4.2.4.2 Value

This parameter shall contain the actual value of the simple data element.

14.4.3 Data Access Error parameter

The Data Access Error parameter shall indicate the reason for failure of an attempted access to a variable. The possible values for this parameter are as follows.

OBJECT-INVALIDATED - An attempted access references a defined object that has an undefined reference attribute. This represents a permanent error for access attempts to that object.

HARDWARE-FAULT - An attempt to access the variable has failed due to a hardware fault.

TEMPORARILY-UNAVAILABLE - The requested variable is temporarily unavailable for the requested access.

Example: A VMD may disallow an attempt to write to the set-point of a control loop that has been placed in manual mode.

OBJECT-ACCESS-DENIED - The MMS client has insufficient privilege to request this operation.

OBJECT-UNDEFINED - The object with the desired name does not exist.

INVALID-ADDRESS - Reference to the unnamed variable object's specified address is invalid because the specified format is incorrect or is out of range.

TYPE-UNSUPPORTED - An inappropriate or unsupported type is specified for a variable.

TYPE-INCONSISTENT - A type is specified that is inconsistent with the service or referenced object.

OBJECT-ATTRIBUTE-INCONSISTENT - The object is specified with inconsistent attributes.

OBJECT-ACCESS-UNSUPPORTED - The variable is not defined to allow requested access.

OBJECT-NON-EXISTENT - The variable is nonexistent.

OBJECT-VALUE-INVALID - The proposed value is not consistent with the set of allowable values for this object.

NOTE The Data Access Error parameter does not indicate failure of a service request. Instead, it indicates failure of an attempted valid access request. The OBJECT-INVALIDATED error indicates that the mapping represented by a reference contained in a defined Named Variable List is no longer valid. All other values indicate failure of the V-Get or V-Put function.

14.5 Specification of Access to Variables

This subclause describes the parameters that specify access to a variable. These include the Variable Access Specification Parameter, the Variable Specification Parameter, and the Address parameter.

14.5.1 Variable Access Specification parameter

The structure of the Variable Access Specification parameter is shown in Table 73.

Table 73 - Variable Access Specification parameter

Parameter Name	Req/Rsp	Ind/Cnf	CBB
Kind of Access	M	M(=)	
List of Variable	S	S(=)	
Variable Specification	M	M(=)	
Alternate Access	U	U(=)	valt
Variable List Name	S	S(=)	vlis

14.5.1.1 Kind Of Access

This parameter shall indicate whether the access is specified in terms of an enumerated list of Variable Specification and (optionally) Alternate Access parameters or in terms of a single Variable List Name parameter giving the value of the &name field of a Named Variable List object.

14.5.1.2 List Of Variable

If Kind Of Access specifies an enumerated list, this parameter shall be specified, otherwise it shall not be specified. If specified, this parameter shall list each variable (one or more) to be accessed, along with any Alternate Access which shall apply. Each element of this list shall contain the following parameters.

14.5.1.2.1 Variable Specification

The Variable Specification parameter shall identify the variable (at the VMD) whose value:

a) is to be read (Read request and indication service primitives);

b) is to be written (Write request and indication service primitives); or

c) has been read (InformationReport service and Read response and confirm service primitives).

The parameters of the Variable Specification parameter are specified in 14.5.2.

14.5.1.2.2 Alternate Access

This parameter shall specify the alternate access that is applicable for this service instance. If not included, full access as specified by the variable's definition is to be used.

This parameter shall be omitted if the variable's Kind of Type is SIMPLE.

14.5.1.3 Variable List Name

If Kind Of Access specifies a named list, this parameter, of type Object Name, shall be specified, otherwise it shall not be specified. If specified this parameter shall provide the value of the Variable List Name attribute of a Named Variable List object at the VMD. This object shall specify a list of one or more variables that:

a) are to be read (Read request and indication service primitives);

b) are to be written (Write request and indication service primitives); or

c) have been read (Read response and confirm service primitives and InformationReport service).

14.5.2 Variable Specification parameter

The Variable Specification parameter shall specify access to a single MMS variable. The structure of its component parameters is shown in Table 74.

Table 74 - Variable Specification parameter

Parameter Name	Req/Rsp	Ind/Cnf	CBB
Kind of Variable	M	M(=)	
Name	S	S(=)	vnam
Address	S	S(=)	vadr
Variable Description	S	S(=)	vadr
Address	M	M(=)	
Type Specification	M	M(=)	

14.5.2.1 Kind Of Variable

This parameter shall indicate the kind of variable access that is to be (or has been) performed. Its value shall be selected from the following:

NAMED - indicating access using a Named Variable object. If this value is selected the Name Parameter shall be present.

UNNAMED - indicating access using an Unnamed Variable object. If this value is specified the Address parameter shall be present.

SINGLE - indicating access using a temporarily created Named Variable object whose definition is supplied in the access request. If this value is specified the Variable Description parameter shall be present.

INVALIDATED - indicating an attempted access to an invalidated variable. This value may only occur in a response or a confirm primitive. If this value is specified, the Name, Address, and Variable Description parameters shall be absent.

14.5.2.2 Name

The Name parameter, of type Object Name, shall specify the &name field of a Named Variable object.

14.5.2.3 Address

The Address parameter shall specify access using the built-in type specified by an Unnamed Variable object. The parameters of the Address parameter are given in 14.5.3.

14.5.2.4 Variable Description

The Variable Description parameter shall specify Address and Type Specification for access using a temporarily created Named Variable object. The created object shall be deleted following the access.

14.5.2.4.1 Address

The Address parameter shall specify the address of the variable being described. It represents the base address of the variable, as described by the Type Description parameter.

The Address parameter is described in 14.5.3.

14.5.2.4.2 Type Specification

The Type Specification parameter shall specify the variable's abstract type. The specified type shall be compatible with the Type Description attributes of all Unnamed Variable objects that are included in the variable. The definition of "compatible" is a local issue. (See description of the Address attribute in 14.1.3). The Type Specification parameter is described in 14.2.

14.5.3 Address parameter

The Address parameter is used to specify an Unnamed Variable object. This International Standard provides three forms of implementation-defined addresses, as shown by the structure of the Address parameter given in Table 75.

Table 75 - Address parameter

Conformance: vadr Parameter Name	Req/Rsp	Ind/Cnf	CBB
Kind of Address	M	M(=)	
Numeric Address	S	S(=)	
Symbolic Address	S	S(=)	
Unconstrained Address	S	S(=)	

14.5.3.1 Kind Of Address

This parameter shall indicate the kind of address contained in the parameter. Its value shall indicate whether the address is a numeric address, a symbolic address or an unconstrained address.

NOTE This International Standard does not specify the relationship between the Kind of Address and the characteristics of a real system. The difference between the various kinds of address, as far as this part of ISO 9506 is concerned, is purely syntactic. An implementation may support zero or more of these kinds of addresses.

14.5.3.2 Numeric Address

This parameter, of type unsigned integer, shall be selected if the Kind Of Address parameter specifies a numeric address. Its use is appropriate for addressing variables in a system that specifies a linear address space or in which addresses can be represented by non-negative integer values.

14.5.3.3 Symbolic Address

This parameter, of type character string, shall be selected if the Kind Of Address parameter specifies a symbolic address. Its use is appropriate for addressing symbolically named built-in variables.

NOTE This kind of address, like the other two, specifies the &address field of an Unnamed Variable. It is semantically and syntactically distinct from an MMS Variable Name.

14.5.3.4 Unconstrained Address

This parameter, of type octetstring, shall be selected if the Kind Of Address parameter specifies an unconstrained address. Its use is appropriate for addressing variables in a system having an implementation-specific address format that may not be represented as a relative address or as a symbolic address.

14.6 Read service

The read service is used by an MMS client in order to request that a MMS server return the value of one or more variables defined at the VMD.

14.6.1 Structure

The structure of the component service primitives is shown in Table 76.

Table 76 - Read service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Specification with Result	M	M(=)			
Variable Access Specification	M	M(=)			
Result(+)			S	S(=)	
Variable Access Specification			C	C(=)	
List of Access Result			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.6.1.1 Argument

This parameter shall convey the service specific parameters for the Read service request.

14.6.1.1.1 Specification With Result

This boolean parameter shall indicate whether (true) or not (false) the Variable Access Specification parameter is requested in the Result(+) parameter of the response primitive, if issued. If true, and the response primitive specifies success, the value of the Variable Access Specification parameter of the indication primitive shall be returned in the Result(+) parameter of the response primitive. If false, the Variable Access Specification parameter shall not be included.

14.6.1.1.2 Variable Access Specification

This parameter shall specify the variables that are to be accessed. The Variable Access Specification parameter is described in 14.5.1.

14.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

NOTE For the Read service, a successful result means that the service request was acceptable to the MMS server and that the MMS server has attempted to determine the value of each of the variables requested by the service.

14.6.1.2.1 Variable Access Specification

This parameter shall be present if requested in the indication primitive. Otherwise, it shall be omitted. If included, it shall contain the Variable Access Specification parameter from the indication primitive.

14.6.1.2.2 List Of Access Result

This parameter shall contain the values of the specified variables, in the order specified by the Variable Access Specification parameter. Each element of the list shall be an Access Result, which shall either specify the value of the real variable at the time of access, after applying the variable's type description and alternate access (if applicable), or a reason for access error. The Access Result parameter is described in 14.4.1.

14.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.6.2 Service Procedure

14.6.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = READ. If the Variable Access Specification parameter specifies a Named Variable List object, the MMS server shall verify that all the conditions in the Access Control List specified by the &accessControl field of the Named Variable List object are satisfied for the service class = READ (see 9.1.3). If these conditions are not satisfied, the service request fails and a Result(-) shall be returned.

14.6.2.2 Actions

For each item of the Variable Specification (whether included in a Named Variable List object or specified individually) the MMS server shall:

a) verify that all the conditions in the Access Control List specified by the &accessControl field of that item are satisfied for the service class = READ. Otherwise, the read operation fails for that component and the Data Access Error OBJECT-ACCESS-DENIED shall be returned as the corresponding component of the Access Result Parameter.

b) attempt to read (see V-Get in 14.1.1.2.1) the value of the specified item and return the value of this item or an Data Access Error describing the problem with this item.

Return a Result(+) with the results of the read operations.

14.7 Write service

The Write service is used by an MMS client to request that the MMS server replace the content of one or more variables with values supplied in the request.

14.7.1 Structure

The structure of the component service primitives is shown in Table 77.

Table 77 - Write service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Variable Access Specification	M	M(=)			
List of Data	M	M(=)			
Result(+)			S	S(=)	
List of Write Result			M	M(=)	
Success			M	M(=)	
Data Access Error			C	C(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.7.1.1 Argument

This parameter shall convey the service specific parameters for the Write service request.

14.7.1.1.1 Variable Access Specification

This parameter shall specify the variable or variables that are to be written. This parameter is described in detail in 14.5.1.

14.7.1.1.2 List Of Data

This parameter shall specify the values to be written to the variables specified by the Variable Access Specification parameter. The values shall occur in this list in the order of the variables specified in the Variable Access Specification parameter. The parameters of Data shall be determined by the variable's type description and alternate access description, as applicable. (See 14.2 and 14.3 for details).

The Data parameter is described in 14.4.2.

14.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

NOTE For the Write service, a successful result means that the service request was acceptable to the MMS server and that the MMS server has attempted to replace the value of each of the specified variables with the values supplied in the request.

14.7.1.2.1 List Of Write Result

The List Of Write Result parameter shall return a list, specified in the order of the variables identified in the request. This list shall indicate, for each variable, either a confirmation that the write to that variable succeeded or a reason that the write to that variable failed.

14.7.1.2.1.1 Success

This boolean parameter shall indicate, for a given variable, whether the write succeeded (true) or not (false).

14.7.1.2.1.2 Data Access Error

If failure is indicated (Success equals false), this parameter shall provide the reason for failure. The description for Data Access Error is found in 14.4.3.

14.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.7.2 Service Procedure

14.7.2.1 Preconditions

The MMS server shall verify the validity of the service request by inspecting the entire Variable Access Specification and the List of Data. If the elements of the List of Data do not agree with the Variable Access Specification in type and number of elements, a Result(-) shall be returned. The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = WRITE. If the Variable Access Specification parameter specifies a Named Variable List object, the MMS server shall verify that all the conditions in the Access Control List specified by the &accessControl field of the Named Variable List object are satisfied for the service class = WRITE (see 9.1.3). If these conditions are not satisfied, the service request fails and a Result(-) shall be returned.

14.7.2.2 Actions

For each item of the Variable Specification (whether included in a Named Variable List object or specified individually) the MMS server shall:

a) verify that all the conditions in the Access Control List specified by the &accessControl field of that item are satisfied. If this condition is not satisfied, a Data Access Error OBJECT-ACCESS-DENIED shall be returned as the corresponding component of the Data Access Error Parameter.

b) attempt to write (see V-Put in 14.1.1.2.2) the value of the specified variable.

The MMS server shall return, in the order specified in the Variable Access Specification parameter, a confirmation for each item that the write succeeded or an indication of why the write failed.

14.8 InformationReport service

The InformationReport service is used by an MMS-user in order to inform the other MMS-user of the value of one or more specified variables, as read by the issuing MMS-user.

14.8.1 Structure

The structure of the component service primitives is shown in Table 78.

Table 78 - InformationReport service

Parameter Name	Req	Ind	CBB
Argument	M	M(=)	
Variable Access Specification	M	M(=)	
List of Access Result	M	M(=)	

14.8.1.1 Argument

This parameter shall convey the service specific parameters for the InformationReport service request.

14.8.1.1.1 Variable Access Specification

This parameter shall identify the variables for which values are being reported. This parameter is fully described in 14.5.1.

14.8.1.1.2 List Of Access Result

This parameter shall contain the values of the specified variables, in the order specified by the Variable Access Specification parameter. Each element of the list shall be an Access Result, which shall either specify the value of the variable at the time of access, or a reason for failure. The Access Result parameter is described in 14.4.1.

14.8.2 Service Procedure

This International Standard does not specify a procedure for invoking, or for receiving, the InformationReport service. The use of this service is application determined. An InformationReport.request shall not be issued if the peer MMS-user did not indicate support of the InformationReport service in the Services Supported parameter in the Initiate service.

This is an unconfirmed service.

NOTE 1 The choice of associations (if more than one exists) on which to send the InformationReport service request is a local matter (that may be further specified by Companion Standards). All associations, one association, or some group may be selected.

NOTE 2 The use of this service is functionally equivalent to an Event Notification with an Event Action of the Read service (with a Specification With Result parameter set to true). The practical difference is that by using the Event Notification method, the conditions under which the service is used are directly visible (and modifiable) using MMS services, while by using the InformationReport service, the conditions are a local matter and cannot be determined or changed by the remote MMS user.

14.9 GetVariableAccessAttributes service

The GetVariableAccessAttributes service is used by an MMS client in order to request that the MMS server return the attributes of a Named Variable or an Unnamed Variable object defined at the VMD.

14.9.1 Structure

The structure of the component service primitives is shown in Table 79.

Table 79 - GetVariableAccessAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Kind of Variable	M	M(=)			
Name	S	S(=)			vnam
Address	S	S(=)			vadr
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Address			C	C(=)	vadr
Type Description			M	M(=)	
Access Control List			C	C(=)	aco
Meaning			C	C(=)	sem
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.9.1.1 Argument

This parameter shall convey the service specific parameters for the GetVariableAccessAttributes service request.

14.9.1.1.1 Kind Of Variable

This parameter shall equal NAMED or UNNAMED depending upon whether the request is for a Named Variable object or an Unnamed Variable object, respectively.

14.9.1.1.2 Name

If Kind Of Variable is equal to NAMED, this parameter, of type Object Name, shall specify the &name field of the desired Named Variable object.

14.9.1.1.3 Address

If Kind Of Variable is equal to UNNAMED, this parameter shall specify the &address field of the desired Unnamed Variable object.

14.9.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

14.9.1.2.1 MMS Deletable

Subclause 9.1.4 specifies the value to be returned by this parameter.

14.9.1.2.2 Address

If the **vadr** parameter conformance building block has been negotiated for the current application association, and if the referenced object is a Named Variable object with &accessMethod field equal to **public**, this parameter shall be the &address attribute of the Named Variable object. Otherwise, this parameter shall be omitted.

14.9.1.2.3 Type Description

This parameter shall be the &typeDescription attribute of the referenced Named Variable or Unnamed Variable object.

14.9.1.2.4 Access Control List

This parameter, of type Identifier, shall be the value of &accessControl field. This field indicates the Access Control List object that controls access to this Variable. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

14.9.1.2.5 Meaning

This parameter, of type character string, shall be the value of the &meaning field, if present. This parameter shall not occur unless the Kind of Variable parameter specifies NAMED, and unless the **sem** CBB has been negotiated.

14.9.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.9.2 Service Procedure

The MMS server shall return the attributes or derived type description, as applicable of the referenced object.

14.10 DefineNamedVariable service

The DefineNamedVariable service is used by an MMS client to request that the MMS server create a Named Variable object that describes a mapping to a real variable in the VMD.

NOTE The ability to define a Named Variable object, using MMS services, has been included in this part of ISO 9506 in order to support those systems that, due to simplicity or age, do not provide for local definition of Named Variable objects.

14.10.1 Structure

The structure of the component service primitives is shown in Table 80.

Table 80 - DefineNamedVariable service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Variable Name	M	M(=)			
Address	M	M(=)			
Type Specification	U	U(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.10.1.1 Argument

This parameter shall convey the service specific parameters for the DefineNamedVariable service request.

14.10.1.1.1 Variable Name

The Variable Name parameter, of type Object Name, shall specify the name that shall uniquely identify the Named Variable object at the VMD. This name shall be unique among Named Variable objects having the specified scope.

14.10.1.1.2 Address

This parameter shall specify the &address field of an Unnamed Variable object. It shall specify the base or starting address for the variable described by the Type Specification parameter. If multiple addresses are required to represent this type, they shall be assigned to contiguous locations of the VMD. The Address parameter is described in 14.5.3.

14.10.1.1.3 Type Specification

This parameter is optional. When specified, it defines the &typeDescription field of the Named Variable object that is being defined. If not specified, the Named Variable object inherits the Unnamed Variable object's &typeDescription field. The Type Specification parameter is described in 14.2.3.

The simple data elements of the type described by the Type Specification shall be compatible with the &typeDescription fields of the Unnamed Variable objects that are spanned by this definition.

NOTE The criteria for deciding that types are compatible are local issues. See description of Address attribute in 14.1.3.

14.10.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

14.10.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.10.2 Service Procedure

14.10.2.1 Preconditions

The MMS server shall verify that no Named Variable object exists that has the same &name field as the Variable Name parameter of the service request. The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD. If these conditions are not satisfied, the service request fails and a Result(-) shall be returned.

14.10.2.2 Actions

The MMS server shall create a Named Variable object and shall initialize its fields as described below.

a) The &name field shall be initialized to the Variable Name parameter.

b) The &accessControl field shall indicate an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) A reference to this newly created Named Variable object shall be added to the &NamedVariables field of the Access Control List object referenced by the &accessControl field of the newly created Named Variable object.

d) If the Type Specification parameter is present in the indication primitive, the Named Variable object's &typeDescription field shall be initialized to the value of the Type Specification parameter, with all Type Name parameters, if any, replaced by the &typeDescription field of the Named Type object having &name field equal to the value of the Type Name parameter.

e) If the Type Specification parameter is absent from the indication primitive, the Named Variable object's &typeDescription field shall be initialized to equal the &typeDescription field of the Unnamed Variable object having &address field equal to the specified Address parameter.

f) The &accessMethod field shall be initialized to **public**.

g) The &address field shall be initialized to the value of the Address parameter.

h) If the Type Specification parameter is present in the indication primitive, and if the Type Name choice has been specified for this parameter, the &meaning field, if present, shall be initialized as follows:

1) if the Scope of the name of the Named Type is VMD-specific, to the value of the &name field of the Named Type;

2) if the Scope of the name of the Named Type is Domain-specific, the Scope of the name of the Named Variable is Domain-specific, and the same Domain is referenced by both objects, to the value of the &name field of the Named Type;

3) if the Scope of the name of the Named Type is AA-specific and the Scope of the name of the Named Variable is AA-specific, to the value of the &name field of the Named Type;

4) otherwise, to a null string.

A Result(+) shall be issued. This response shall contain no service-specific information.

14.11 DeleteVariableAccess service

The DeleteVariableAccess service is used by an MMS client to request that the MMS server delete one or more Named Variable objects for which deletion is permitted.

14.11.1 Structure

The structure of the component service primitives is shown in Table 81.

Table 81 - DeleteVariableAccess service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
List of Variable List Name	C	C(=)			
Domain Name	C	C(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Number Deleted			M	M(=)	

14.11.1.1 Argument

This parameter shall convey the service specific parameters for the DeleteVariableAccess service request.

14.11.1.1.1 Scope of Delete

The Scope of Delete parameter shall specify the extent of delete to be attempted. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the specific Named Variable objects having &name fields equal to the Name parameters of the List of Name parameter for which deletion is permitted are to be deleted.

AA-SPECIFIC - Specifies that all Named Variable objects within the scope of the current application association for which deletion is permitted are to be deleted.

DOMAIN - Specifies that all Named Variable objects within the scope of the specified Domain for which deletion is permitted are to be deleted.

VMD - Specifies that all Named Variable objects having VMD scope for which deletion is permitted are to be deleted.

14.11.1.1.2 List Of Name

This parameter shall be specified if Scope of Delete is SPECIFIC. Otherwise, it shall be omitted. If included, it shall contain a list of one or more Name parameters, of type Object Name, each specifying the &name attribute of a Named Variable object that is to be deleted.

14.11.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if Scope of Delete is equal to DOMAIN. Otherwise, it shall be omitted. It provides the name of the Domain for which all Named Variable objects for which deletion is permitted are to be deleted.

14.11.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

14.11.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of Named Variable objects that matched the name specification in the service request.

14.11.1.2.2 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable objects that were deleted as a result of executing the service procedure.

NOTE The difference between the Number Matched and Number Deleted parameters indicate the number of objects that were not deleted, either because the conditions specified in the referenced Access Control List object(s) were not satisfied for Service Class = DELETE, or for other reasons.

14.11.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

14.11.1.3.1 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable objects that were deleted as a result of executing the service procedure.

14.11.2 Service Procedure

14.11.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &aAccessControl field of the VMD are satisfied for the service class = DELETE. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

14.11.2.2 Actions

The MMS server shall prepare a list of objects to be deleted. If SPECIFIC was selected as the Scope of Delete parameter, the List of Name parameter identifies the objects to be deleted. Otherwise, all Named Variable objects of the indicated scope are to be deleted. For each object on the list, the conditions in the Access Control List specified by the &accessControl field of the object to be deleted shall be evaluated. If the conditions are satisfied, and if the deletion is otherwise fasible, the MMS server shall perform the following steps.

a) The MMS server shall remove the reference to this object from the &NamedVariables field of the Access Control List object referenced by the &accessControl field of this object.

b) If a deleted object is referenced by the &namedItem field of one or more Named Variable List objects, each deleted object shall be removed from the corresponding &namedItem field.

c) The MMS server shall delete the object.

If the conditions are not satisfied, do not delete this object.

After all objects of the specified scope have been deleted, a Result(+) shall be issued with the values assigned to the Number Matched and Number Deleted parameters.

If an error occurs in the deletion of any of the specified objects, a Result(-) shall be issued with the Number Deleted parameter indicating the number of objects that were deleted. Failure to delete an object because the conditions specified in the referenced Access Control List object were not satisfied shall not be deemed an error.

14.12 DefineNamedVariableList service

The DefineNamedVariableList service is used by an MMS client to request that the MMS server create a Named Variable List object.

14.12.1 Structure

The structure of the component service primitives is shown in Table 82.

Table 82 - DefineNamedVariableList service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Variable List Name	M	M(=)			
List of Variable	M	M(=)			
Variable Specification	M	M(=)			
Alternate Accesss	U	U(=)			valt
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.12.1.1 Argument

This parameter shall convey the service specific parameters for the DefineNamedVariableList service request.

14.12.1.1.1 Variable List Name

The Variable List Name parameter, of type Object Name, shall specify the name that shall uniquely identify the Named Variable List object at the VMD.

14.12.1.1.2 List Of Variable

The List Of Variable parameter shall specify a list of one or more variables that are to be accessed using the Named Variable List. Each element of this list shall specify the following parameters.

14.12.1.1.2.1 Variable Specification

This parameter shall specify a variable that is to be accessed by this element of the list. The Variable Specification parameter is described in 14.5.2.

14.12.1.1.2.2 Alternate Access

This optional parameter, if included, shall specify Alternate Access that is to apply when the variable specified by this element of the list is accessed. If omitted, full access is specified. Alternate Access is described in 14.3.

14.12.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

14.12.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.12.2 Service Procedure

14.12.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

14.12.2.2 Actions

The MMS server shall create a Named Variable List object and shall initialize its attributes as follows:

a) The &name field shall be initialized to equal the Variable List Name parameter.

b) The &accessControl field shall indicate an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) A reference to this newly created Named Variable List object shall be added to the &NamedVariableLists field of the Access Control List object referenced by the &accessControl field of the newly created Named Variable List object.

d) The &listOfVariables field shall be initialized to identify variable objects and access descriptions for those variable objects. The entries shall be maintained in the Named Variable List object in the order that they occur in the List Of Variable parameter. Each entry shall correspond to an element of the List Of Variable parameter and shall have its attributes initialized as specified below.

   1) If the Kind Of Variable parameter of the Variable Specification parameter is NAMED, a &namedItem field shall be created, indicating the Named Variable identified in the Variable Specification parameter. If the Alternate Access parameter is present for this item, the &accessMethod field shall be set to this parameter; otherwise this field shall be omitted.

   2) If the Kind Of Variable parameter of the Variable Specification parameter is UNNAMED, an &unnamedItem field shall be created, indicating the Unnamed Variable identified in the Variable Specification parameter. If the Alternate Access parameter is present for this item, the &accessMethod field shall be set to this parameter; otherwise this field shall be omitted.

   3) If the Kind of Variable parameter of the Variable Specification parameter is SINGLE, a Named Variable object shall be created with its &name field equal to UNDEFINED, its &typeDescription field equal to the type specified by the Variable Description parameter's Type Specification, its &accessMethod field equal to **public**, and its &address field equal to the Address of the Variable Specification parameter. Its &accessControl field shall be set to reference an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose. If the Alternate Access parameter is present for this item, the &accessMethod field shall be set to this parameter; otherwise this field shall be omitted.

e) A Result(+) shall be issued.

14.13 GetNamedVariableListAttributes service

The GetNamedVariableListAttributes service is used by an MMS client to request that an MMS server return the attributes of a Named Variable List object defined at the VMD.

14.13.1 Structure

The structure of the component service primitives is shown in Table 83.

Table 83 - GetNamedVariableListAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Variable List Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
List of Variable			M	M(=)	
Variable Specification			M	M(=)	
Alternate Access			C	C(=)	valt
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.13.1.1 Argument

This parameter shall convey the service specific parameters for the GetNamedVariableListAttributes service request.

14.13.1.1.1 Variable List Name

The Variable List Name parameter, of type Object Name, shall specify the &name field of the Named Variable List object whose attributes are desired.

14.13.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

14.13.1.2.1 MMS Deletable

This parameter shall indicate whether (true) or not (false) the Named Variable List object is deletable using the DeleteNamedVariableList service. Subclause 9.1.4 specifies the value to be returned by this parameter.

14.13.1.2.2 List Of Variable

The List Of Variable parameter shall be the &listOfVariable field of the Named Variable List object. It shall specify an ordered list of one or more access descriptions, each specifying the Variable Specification and (conditionally) Alternate Access parameters that specify access to an MMS variable object.

14.13.1.2.2.1 Variable Specification

The Variable Specification parameter shall specify the variable that is accessed by this element of the list. Variable Specification is described in 14.5.2.

14.13.1.2.2.2 Alternate Access

This parameter shall be omitted if full access to the referenced variable is provided. Otherwise, it shall specify the value of the Access Description attribute of this list element. Alternate Access is described in 14.3.

14.13.1.3 Access Control List

This parameter, of type Identifier, shall indicate the name of the Access Control List object that controls access to this Named Variable List object. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

14.13.1.4 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.13.2 Service Procedure

The MMS server shall return the attributes of the Named Variable List object.

Entries shall be placed in the List Of Variable parameter in the order specified in the &listOfVariables field of the Named Variable List object. Each entry shall correspond to an element of the Named Variable List object's &listOfVariable field and shall have its parameters specified as follows:

a) The parameters of the Variable Specification parameter shall be determined by the choice made of item type within the &listOfVariables field.

   1) If the choice is &namedItem and the referenced Named Variable object's &name field does not have the value UNDEFINED, the Kind Of Variable parameter shall be NAMED and Name shall contain the &name field of the referenced object.

   2) If the choice is &unnamedItem, the Kind Of Variable parameter shall be UNNAMED, and Address shall contain the &address field of the referenced object.

   3) If the choice is &namedItem and the referenced Named Variable object's &name field is equal to UNDEFINED, the Kind Of Variable parameter shall be SINGLE and Variable Description shall contain Address equal to the &address field of the referenced Named Variable object and Type Specification equal to the referenced Named Variable object's &typeDescription field.

b) The Alternate Access parameter shall be omitted if the &accessMethod attribute is missing. Otherwise, it shall be equal to the &accessMethod attribute.

A Result(+) shall be issued.

14.14 DeleteNamedVariableList service

The DeleteNamedVariableList service is used by an MMS client to request that an MMS server delete one or more MMS-defined Named Variable List objects at the VMD whose deletion is allowed.

14.14.1 Structure

The structure of the component service primitives is shown in Table 84.

Table 84 - DeleteNamedVariableList service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
List of Variable List Name	C	C(=)			
Domain Name	C	C(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Number Deleted			M	M(=)	

14.14.1.1 Argument

This parameter shall convey the service specific parameters for the DeleteNamedVariableList service request.

14.14.1.1.1 Scope of Delete

The Scope of Delete parameter shall specify the extent of deletes to be attempted. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the specific Named Variable List objects having &name fields equal to the &name field of the Named Variable List objects indicated by the List Of Variable List Name parameter are to be deleted.

AA-SPECIFIC - Specifies that all MMS-defined Named Variable List objects within the scope of the current application association are to be deleted.

DOMAIN - Specifies that all MMS-defined Named Variable List objects within the scope of the specified Domain are to be deleted.

VMD - Specifies that all MMS-defined Named Variable List objects having VMD scope are to be deleted.

14.14.1.1.2 List Of Variable List Name

This parameter shall be specified if Scope of Delete is SPECIFIC. Otherwise, it shall be omitted. If included, it shall contain a list of one or more Variable List Name parameters, of type Object Name, each specifying the value of the &name attribute of a specific Named Variable List object that is to be deleted.

14.14.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if Scope of Delete is equal to DOMAIN. Otherwise, it shall be omitted. It provides the name of the Domain for which all Named Variable List objects are to be deleted.

14.14.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

14.14.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of Named Variable List objects that matched the name specification in the service request.

14.14.1.2.2 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable List objects that were deleted as a result of executing the service procedure.

14.14.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

14.14.1.3.1 Number Deleted

This parameter, of type integer, shall indicate the number of Named Variable List objects that were deleted as a result of executing the service procedure.

14.14.2 Service Procedure

14.14.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

14.14.2.2 Actions

The MMS server shall prepare a list of objects to be deleted. If SPECIFIC was selected as the Scope of Delete parameter, the List of Variable List Name parameter identifies the objects to be deleted. Otherwise, all Named Variable List objects of the indicated scope are to be deleted. For each object on the list, the conditions in the Access Control List specified by the &accessControl field of the object to be deleted shall be evaluated. If the conditions are satisfied, and if the deletion is otherwise fasible, the MMS server shall perform the following steps.

a) The MMS server shall remove the reference to this object from the &NamedVariableLists field of the Access Control List object referenced by the &accessControl field of this object.

b) If a deleted Named Variable List object references a Named Variable object having its &name field equal to UNDEFINED, the referenced object shall also be deleted. Any such referenced objects deleted shall not be included in the count of the number matched or the number deleted.

c) The MMS server shall delete the Named Variable List object.

If the conditions are not satisfied, do not delete this object.

After all Named Variable List objects of the specified scope have been deleted, a Result(+) shall be issued with the values assigned to the Number Matched and Number Deleted parameters.

If an error occurs in the deletion of any of the specified objects, a Result(-) shall be issued with the Number Deleted parameter indicating the number of objects that were deleted. Failure to delete an object for which the conditions in the Access Control List object referenced by the &accessControl field are not satisfied for service class = DELETE shall not be deemed an error.

14.15 DefineNamedType service

The DefineNamedType service is used by an MMS client to request that an MMS server store a type description for use in subsequent definition of Named Variable or Named Type (or both) objects at the VMD.

14.15.1 Structure

The structure of the component service primitives is shown in Table 85.

Table 85 - DefineNamedType service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Type Name	M	M(=)			
Type Specification	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.15.1.1 Argument

This parameter shall convey the service specific parameters for the DefineNamedType service request.

14.15.1.1.1 Type Name

The Type Name parameter, of type Object Name, shall specify the name that shall uniquely identify the Named Type object at the VMD.

14.15.1.1.2 Type Specification

This parameter shall specify the value of the abstract type that shall be associated with the Type Name. The Type Specification parameter is described in 14.2.3.

14.15.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

14.15.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.15.2 Service Procedure

14.15.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

14.15.2.2 Actions

The MMS server shall create a Named Type object and shall initialize its fields as described below.

a) The &name field shall be initialized to equal the Type Name parameter.

b) The &accessControl field shall indicate an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) A reference to this newly created Named Type object shall be added to the &NamedTypes field of the Access Control List object referenced by the &accessControl field of the newly created Named Type object.

d) The Named Type object's &typeDescription field shall be initialized to the value of the Type Specification parameter, with all Type Name parameters, if any, replaced by the &typeDescription field of the Named Type object having &name field equal to the value of the Type Name parameter.

e) If the Type Specification parameter is present in the indication primitive, and if the Type Name choice has been specified for this parameter, the &meaning field shall be initialized as follows. In this description the defined Named Type refers to the Named Type specified by the Type Name parameter of the service indication. The defining Named Type refers to the Named Type choice of the Type Specification parameter. The &meaning field of the defined Named Type shall be initialized as follows:

   1) if the Scope of the name of the defining Named Type is VMD-specific, to the value of the &name field of the defining Named Type;

   2) if the Scope of the name of the defining Named Type is Domain-specific, the Scope of the name of the defined Named Type is Domain-specific, and the same Domain is referenced by both objects, to the value of the &name field of the defining Named Type;

   3) if the Scope of the name of the defining Named Type is AA-specific and the Scope of the name of the defined Named Variable is AA-specific, to the value of the &name field of the Named Type;

   4) otherwise, to a null string.

A Result(+) shall be issued. This response shall contain no service-specific information.

14.16 GetNamedTypeAttributes service

The GetNamedTypeAttributes service is used by an MMS client to request that an MMS server return the attributes of a Named Type object.

14.16.1 Structure

The structure of the component service primitives is shown in Table 86.

Table 86 - GetNamedTypeAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Type Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Type Description			M	M(=)	
Access Control List			C	C(=)	aco
Meaning			C	C(-)	sem
Result(-)			S	S(=)	
Error Type			M	M(=)	

14.16.1.1 Argument

This parameter shall convey the service specific parameters for the GetNamedTypeAttributes service request.

14.16.1.1.1 Type Name

The Type Name parameter, of type Object Name, shall specify the value of the &name field of the Named Type object whose attributes are desired.

14.16.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

14.16.1.2.1 MMS Deletable

This parameter shall indicate whether (true) or not (false) the Named Type object was may be deleted using the DeleteNamedType service. Subclause 9.1.4 specifies the value to be returned by this parameter.

14.16.1.2.2 Type Description

This parameter shall contain the value of the &typeDescription field of the Named Type object. The Type Description parameter is described in 14.2.2.

14.16.1.2.3 Access Control List

This parameter, of type Identifier, shall indicate the name of the Access Control List object that controls access to this Named Type. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

14.16.1.2.4 Meaning

This parameter, of type character string, shall be the value of the &meaning field, if present. This parameter shall not occur unless the **sem** CBB has been negotiated.

14.16.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

14.16.2 Service Procedure

The MMS server shall return the parameters associated with this Named Type object. Subclause 9.1.4 specifies the value to be returned by the MMS Deletable parameter.

14.17 DeleteNamedType service

The DeleteNamedType service is provided in order to allow an MMS client to request that an MMS server delete one or more Named Type objects.

14.17.1 Structure

The structure of the component service primitives is shown in Table 87.

Table 87 - DeleteNamedType service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
List Of Type Name	C	C(=)			
Domain Name	C	C(=)			
Result(+)			S	S(=)	
Number Matched			M	M(=)	
Number Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Number Deleted			M	M(=)	

14.17.1.1 Argument

This parameter shall convey the service specific parameters for the DeleteNamedType service request.

14.17.1.1.1 Scope of Delete

The Scope of Delete parameter shall specify the extent of delete to be attempted. Possible values for this parameter, and their meaning, are as follows:

SPECIFIC - Specifies that the specific Named Type objects having &name field equal to the Type Name parameters of the List Of Type Name parameter are to be deleted.

AA-SPECIFIC - Specifies that all Named Type objects within the scope of the current application association are to be deleted.

DOMAIN - Specifies that all Named Type objects within the scope of the specified Domain are to be deleted.

VMD - Specifies that all Named Type objects having VMD scope are to be deleted.

14.17.1.1.2 List Of Type Name

This parameter shall be specified if Scope of Delete is SPECIFIC. Otherwise, it shall be omitted. If included, it shall contain a list of one or more Type Name parameters, of type Object Name, each specifying the value of the Type Name attribute of a specific Named Type object that to be deleted.

14.17.1.1.3 Domain Name

This parameter, of type Identifier, shall be specified if Scope of Delete is equal to DOMAIN. Otherwise, it shall be omitted. It shall be the name of the Domain for which all Named Type objects are to be deleted.

14.17.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

14.17.1.2.1 Number Matched

This parameter, of type integer, shall indicate the number of Named Type objects that matched the name specification in the service request.

14.17.1.2.2 Number Deleted

This parameter, of type integer, shall indicate the number of Named Type objects that were deleted as a result of executing the service procedure.

14.17.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

14.17.1.3.1 Number Deleted

This parameter, of type integer, shall indicate the number of Named Type objects that were deleted as a result of executing the service procedure.

14.17.2 Service Procedure

14.17.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

14.17.2.2 Actions

The MMS server shall prepare a list of objects to be deleted. If SPECIFIC was selected as the Scope of Delete parameter, the List of Name parameter identifies the objects to be deleted. Otherwise, all Named Type objects of the indicated scope are to be deleted. For each object on the list, the conditions in the Access Control List specified by the &accessControl field of the object to be deleted shall be evaluated. If the conditions are satisfied, and if the deletion is otherwise fasible, the MMS server shall perform the following steps.

a) The MMS server shall remove the reference to this Named Type object from the &NamedTypes field of the Access Control List object referenced by the &accessControl field of this Named Type object.

b) The MMS server shall delete the Named Type object.

If the conditions are not satisfied, do not delete this object.

After all Named Type objects of the specified scope have been deleted, a Result(+) shall be issued with the values assigned to the Number Matched and Number Deleted parameters.

If an error occurs in the deletion of any of the specified objects, a Result(-) shall be issued with the Number Deleted parameter indicating the number of objects that were deleted. Failure to delete an object because the conditions specified in the referenced Access Control List object were not satisfied shall not be deemed an error.

14.18 Conformance

The Variable Access Services define parameter conformance requirements for the MMS server. These requirements are described below.

14.18.1 Static Conformance

The static conformance statement for an implementation shall state the level of support that it provides, if any, for uninterruptibility of access to variables described by Named Variable and Unnamed Variable objects. Uninterruptibility of access is defined to mean that:

a) read access is performed with the guarantee that the variable is not changing while being read;

b) write access is performed with the guarantee that all potential concurrent access to the variable (whether local or remote, for read or for write) is inhibited during the write.

If an implementation supports uninterruptibility of access for a subset of its variables, the static conformance statement shall specify the requirements associated with this subset, and shall specify whether or not variables failing to meet these requirements are accessible using MMS services.

In addition to the required statement concerning uninterruptibility of access, an implementation supporting **vadr** shall specify its address format or formats, and shall specify the relationship between an address and an inherent type associated with that address.

14.19 Guidance To Implementors

Subclause 14.19 and its subclauses are informative.

The MMS Variable Access services are intended to serve the needs of a wide range of systems, from the most simple to the highly complex. These services are also intended to serve the needs of devices designed with this part of ISO 9506 in mind, as well as the needs of devices that, due to age or simplicity, were not designed with this part of ISO 9506 in mind.

Due to this wide range of requirements, not all of the Variable Access services are appropriate for every VMD. It is expected that there will be three primary environments for implementation of MMS Variable access services. They have been classified as follows:

a) **vnam**-only - serving the needs of systems designed with MMS in mind;

b) **vadr**-only - serving the needs of simple systems and of systems that, due to age or other consideration, do not justify **vnam** support;

c) **vnam**-with-**vadr** - serving the needs of systems that were not designed with MMS in mind, and for which it is justified, for whatever reason, to bring as much as possible of the **vnam**-only functionality to the device.

NOTE Other environments are possible and are not prohibited.

The conformance issues associated with these three environments are described below.

14.19.1 VNAM-only

A **vnam**-only system is one that has been designed with MMS in mind. It directly supports the Variable Access model through the local definition of MMS Named Variable objects by the serving application process in which the VMD exists. These objects have Access Method attribute not equal to **public**. Unnamed Variable objects are not supported, though the access method associated with a Named Variable object may, in fact, make use of address information.

From a parameter conformance stand-point, this system will support **vnam**. It is likely to support both **str1** and **str2** (and thus **nest** greater than zero) and it may also support **valt**.

14.19.2 VADR-only

The **vadr**-only system is one that is either quite simple, thus not justifying **vnam**-only, or it has not been designed with MMS in mind and does not justify the complexity of the **vnam**-with-**vadr** environment. This system supports Unnamed Variable objects only. From a parameter conformance stand-point, this system will support **vadr**. It is unlikely to support either **str1** or **str2** (and thus **nest** will be equal to zero and **valt** will not be supported). **vnam** is not supported.

NOTE Note that while no variable names are supported in this system, names for other objects, such as Domains and Program Invocations, may still be supported.

14.19.3 VNAM-with-VADR

This environment encompasses all MMS-capable systems that implement the Variable Access services and that are not included in the previous environments. The goal of this environment should be to present a **vnam**-only view to most clients.

15 Data Exchange Management Services

This clause provides an object model for the following object:

DATA-EXCHANGE

This clause specifies the following services:

ExchangeData
GetDataExchangeAttributes

The Data Exchange management services provide facilities that allow an MMS client to invoke a procedure at the VMD. This remote procedure call is modeled as an exchange of data between two MMS users. Subclause 15.1 describes the MMS Model for Data Exchange. The ExchangeData service is described in 15.2 and the GetDataExchangeAttributes service is described in 15.3. These services are intended to provide functions not available through use of other MMS services, and shall not be used to circumvent the spirit or intent of those services.

15.1 The Data Exchange management model

This clause defines the MMS Model for the Data Exchange object and the related Data Exchange function. A data exchange object is an abstract element of a VMD that is capable of invoking (when requested) a real procedure. This procedure may require data as input and may produce data as output.

NOTE Implementation of Data Exchange on a real device may take different forms. One form could be realisation as a remote procedure call. Another form could be message function blocks on a programmable device. These may be used for synchronization of processing by having a logical thread of execution stalled waiting for receipt of the Data Exchange message.

15.1.1 D-Exchange Function

The D-Exchange function represents the processing of a data exchange service at the VMD. Parameters of the D-Exchange function are the state of the VMD and the values of the input parameters. The relationship between the real procedure and the Data Exchange object that is used to invoke it is modeled by the D-Exchange function. If the processing is successful, the result is a set of values for the output parameters.

15.1.2 The Data Exchange object model

```
DATA-EXCHANGE ::= CLASS {
    &name                   Identifier,
--   this field shall be unique among all Data Exchange objects
    &inUse                  BOOLEAN,
    &accessControl          ACCESS-CONTROL-LIST,
    &request                SEQUENCE OF TypeDescription,
    &response               SEQUENCE OF TypeDescription,
    &linked                 BOOLEAN,
-- The following attribute shall appear if an only if
-- the value of &linked is true.
    &programInvocation      PROGRAM-INVOCATION }
```

15.1.2.1 &name

The &name field uniquely identifies the Data Exchange object within the VMD.

15.1.2.2 &inUse

The &inUse field indicates whether (true) or not (false) the Data Exchange object is performing the D-Exchange function.

15.1.2.3 &accessControl

The &accessControl field identifies an Access Control List object that provides conditions under which this Data Exchange object may be written or have its access control changed.

15.1.2.4 &request

The &request field specifies the data types of input data (if any) to the underlying procedure.

15.1.2.5 &response

The &response field specifies the data types of output data (if any) to the underlying procedure.

15.1.2.6 &linked

The &linked field indicates whether (true) or not (false) the Data Exchange object is linked to a Program Invocation.

15.1.2.7 &programInvocation

The &programInvocation field, which exists only for a Data Exchange object having &linked field equal to true, specifies the Program Invocation to which the Data Exchange object is linked.

15.2 ExchangeData service

The Exchange Data Service is used by an MMS client to invoke a predefined procedure referenced by a Data Exchange object at the VMD.

15.2.1 Structure

The structure of the component service primitives is shown in Table 88.

Table 88 - DataExchange service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Data Exchange Name	M	M(=)			
Result(+)			S	S(=)	
In Use			M	M(=)	
List Of Request Type Specifications			M	M(=)	
List of Response Type Specifications			M	M(=)	
Program Invocation			C	C(=)	
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

15.2.1.1 Argument

This parameter shall contain the parameters of the Exchange Data service request.

15.2.1.1.1 Data Exchange Name

This parameter, of type Object Name, shall specify the &name field of the Data Exchange object that is to be invoked.

15.2.1.1.2 List of Request Data

This parameter, of type List of Data, shall specify the list of data values to be delivered to the Data Exchange object. The data values shall correspond in type and number to that specified by the &request field of the Data Exchange object specified by the Data Exchange Name parameter. If no data values are specified by this List of Request Data parameter, an empty list shall be transmitted.

15.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

15.2.1.2.1 List of Response Data

This parameter, of type List of Data, shall specify the list of data values to be returned to the MMS client. The data values shall correspond in type and number to that specified by the &response field of the Data Exchange object specified by the Data Exchange Name parameter. If no data values are specified by this List of Response Data parameter, an empty list shall be transmitted.

15.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

15.2.2 Service Procedure

15.2.2.1 Preconditions

The MMS Server shall verify:

a) that the Data Exchange object referenced does exist.

b) that the list of request data in the service request conforms to the data types specified in the &request field of the Data Exchange object.

c) that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = WRITE.

d) all the conditions in the Access Control List specified by the &accessControl field of the Data Exchange object are satisfied for the service class = WRITE.

e) if the Data Exchange object is linked to a Program Invocation, that the associated Program Invocation does exist and is in the **running** state. The effect of this service on the Program Invocation, if any, shall be a local matter.

If any of these conditions is not satisfied, the MMS Server shall return a Result(-).

15.2.2.2 Actions

The MMS Server shall set the value of the &inUse field of the Data Exchange object to true and perform a data exchange (see 15.1.1). After the D-Exchange function completes, the MMS Server shall change the value of the &inUse field to false and issue a Result(+) service response, which shall convey the output values of the D-Exchange function in the List of Response Data. The list of data values conveyed in the List of Response Data parameter shall conform in type and number to the &response field of the Data Exchange object.

If the Data Exchange object is linked to a Program Invocation and the Program Invocation transitions out of the **running** state while the Data Exchange function is being performed, success or failure of the service is a local matter.

The method of implementation of the D-Exchange function is a local matter. If the D-Exchange function allows multiple concurrent instances of execution of the procedure, the value of the &inUse field shall be true if any instance is active.

Success or failure of this service shall not be conditioned on the result of the procedure. Results of the procedure, if any, shall be returned as part of List of Response Data.

15.3 GetDataExchangeAttributes service

The GetDataExchangeAttributes service is used by an MMS client to request that an MMS server return the attributes associated with the specified Data Exchange object.

15.3.1 Structure

The structure of the component service primitives is shown in Table 96.

Table 89 - GetDataExchangeAttributes service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Data Exchange Name	M	M(=)			
Result(+)			S	S(=)	
In Use			M	M(=)	
List Of Request Type Specifications			M	M(=)	
List of Response Type Specifications			M	M(=)	
Program Invocation			C	C(=)	
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

15.3.1.1 Argument

This parameter shall contain the parameter of the GetDataExchangeAttributes service request.

15.3.1.1.1 Data Exchange Name

This parameter, of type Object Name, shall be the &name field of the Data Exchange object whose attributes are requested.

15.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

15.3.1.2.1 In Use

This parameter, of type boolean, shall indicate the value of the &inUse field.

15.3.1.2.2 List of Request Type Specifications

This parameter, of type List of Type Specification, shall indicate the value of the &request field. If no types are specified by this List of Request Type Specifications, an empty list shall be transmitted.

15.3.1.2.3 List of Response Type Specifications

This parameter, of type List of Type Specification, shall indicate the value of the &response field. If no types are specified by this List of Response Type Specifications, an empty list shall be transmitted.

15.3.1.2.4 Program Invocation

This optional parameter, of type Identifier, shall indicate the value of the &programInvocation field if present. If the value of the &linked field is true, this parameter shall be present, otherwise it shall be omitted.

15.3.1.2.5 Access Control List

This parameter, of type Identifier, shall be the value of &accessControl field. This field indicates the Access Control List object that controls access to this Data Exchange object. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

15.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

15.3.2 Service Procedure

The MMS server shall verify that the specified Data Exchange object exists. If the Data Exchange object does not exist a Result(-) response shall be returned. Otherwise the attributes of the Data Exchange object shall be returned.

16 Semaphore Management Services

This clause provides object models for the following objects:

SEMAPHORE
SEMAPHORE-ENTRY
NAMED-TOKEN

This clause specifies the following services:

TakeControl
RelinquishControl
DefineSemaphore
DeleteSemaphore
ReportSemaphoreStatus
ReportPoolSemaphoreStatus
ReportSemaphoreEntryStatus
AttachToSemaphore modifier

The Semaphore Management clause contains services that allow synchronization, control and coordination of shared resources between MMS users. The mechanism provided by the MMS server is a set of operations on a semaphore object.

A semaphore may be used by MMS users to control the access to a specific subset of the (real) resources of the MMS server, to control access to MMS objects within the VMD, or to synchronize remote applications. The use of semaphores to control access to MMS objects is described in clause 9. The MMS server enforces these rules with regard to MMS objects. When a semaphore is used to control access to real resources, the MMS server does not enforce the protection of a specific subset by a semaphore and the rules of use of a semaphore in a given application are subject to prior agreement between the user applications.

The application may prevent deadlock by using timers, either in the MMS server by providing a parameter in the service primitive used for obtaining the control of a semaphore, or in the MMS client by cancelling pending requests.

16.1 The Semaphore Management Model

The Semaphore Management Services apply to semaphore objects managed by an MMS server. MMS has two classes of semaphore objects:

a) Token Semaphores;

b) Pool Semaphores.

Each class of semaphore is defined by a state machine and specific attributes.

A semaphore is defined as a specific instantiation of the attributes of a generic semaphore class, and is referenced by a VMD-specific name. This name follows the same rules as every object name in the VMD. The instantiation of a semaphore may be predefined in the VMD or a DefineSemaphore service.

NOTE MMS only provides a service for the definition of token semaphores. A pool semaphore requires a linking between some physical or logical local entities in the real device and the named-token handled by the pool semaphore in the VMD. Typically, a pool semaphore is the representation in the VMD of a real pool semaphore controlling real resources either from local or remote access; MMS cannot create objects in the real devices (only in the VMD), and cannot create an explicit mapping between real and virtual

resources. For this reason, pool semaphores can only be predefined. The creation of a token semaphore is provided by MMS with the intent that this semaphore will be used to synchronize applications between MMS-users, either by prior agreement or by use of the facilities described in clause 9.

A semaphore is modelled as a queue processor, a list of owners and a queue of requesters. Each element of the queue is an object called a semaphore-entry: it is created by a TakeControl request, or by any MMS service request modified by the AttachToSemaphore Modifier, or by local means, and is initialized by the parameters provided in the request. As soon as an element of the queue is served, it is moved into the list of the owners. The two classes of semaphores follow this general model, but each class defines a specific state machine and a specific queue-serving algorithm.

The owner of a semaphore is an Application Process, including possibly an Application Process local to the VMD. The owner is identified by an Application Reference. Unless the owner is a local Application Process, the owner shall maintain the Application Association with the MMS server used for requesting the semaphore until the owner relinquishes control of the semaphore. If the Application Association disappears while a semaphore is still under control of some external Application Process, the control of the semaphore is either relinquished or the semaphore enters the **hung** state, depending upon the parameters specified by the MMS client in the service request to take control of the semaphore.

An MMS client may issue multiple control requests on the same semaphore under the same or multiple Application Associations and gain multiple ownership of a token or pool semaphore. The MMS server is able to differentiate these ownerships as long as they are not under the same Application Association; however, a third party MMS-user using the MMS services cannot differentiate them since the Application Association is known only by the peer entities.

NOTE Multiple ownerships of a pool semaphore under the same association can be differentiated by the named-token; multiple ownerships of a token semaphore under the same association can not be differentiated.

16.1.1 The Semaphore Object

This clause introduces the model of a semaphore object.

```
SEMAPHORE ::= CLASS {
    &name                   Identifier,
-- shall be unique among all semaphores within the VMD
    &accessControl          ACCESS-CONTROL-LIST,
    &class                  ENUMERATED {
        token                   (0),
        pool                    (1) },
-- If the value of &class is token, the following two fields shall appear
    &numberOfTokens         INTEGER OPTIONAL,
    &numberOfOwnedTokens    INTEGER OPTIONAL,
-- If the value of &class is pool, the following field shall appear
    &NamedTokens            NAMED-TOKEN OPTIONAL,
    &Owners                 SEMAPHORE-ENTRY,
    &Requesters             SEMAPHORE-ENTRY,
    &eventCondition         EVENT-CONDITION }
```

16.1.1.1 &name

The &name field identifies the semaphore. It shall be a VMD-specific Object Name.

16.1.1.2 &accessControl

The &accessControl field identifies an Access Control List object that provides conditions under which this semaphore may be controlled by a MMS client, deleted, or have its access control changed.

16.1.1.3 &class

The &class field may have the value **token** or **pool**; it specifies the class of the semaphore.

16.1.1.4 &numberOfTokens

The &numberOfTokens field defines the maximum number of owners allowed for a token semaphore.

16.1.1.5 &numberOfOwnedTokens

The &numberOfOwnedTokens field contains the number of tokens currently owned for a token semaphore.

16.1.1.6 &NamedTokens

The &NamedTokens field identifies a set of named-token objects controlled by a pool semaphore.

16.1.1.7 &Owners

The &Owners field identifies a set of semaphore-entry objects owning this semaphore.

16.1.1.8 &Requesters

The &Requesters field identifies a set of semaphore-entry objects waiting to obtain the control of this semaphore.

16.1.1.9 &eventCondition

The &eventCondition field identifies an Event Condition object whose &name field value is equal to the semaphore object's &name field value, whose &accessControl field references an Access Control List object that specifies **never** for the choice of AccessCondition in the &deleteAccessCondition field, whose &eClass field value is **network-triggered**, whose &ecState field value is **disabled**, whose &priority field value is **normalPriority**, and whose &severity field value is **normalSeverity**.

16.1.2 Named-token

This clause introduces the model of the named-token object.

```
NAMED-TOKEN ::= CLASS {
    &name           Identifier,
    &state          ENUMERATED {
        free            (0),
        owned           (1) } }
```

16.1.2.1 &name

The &name field identifies the named-token. It shall be an Identifier and be unique to the semaphore that owns it.

16.1.2.2 &state

The &state field identifies whether the named-token is available to a client (**free**) or is owned by some semaphore-entry object for a client (**owned**).

16.1.3 The Semaphore-entry object

This clause introduces the model of the semaphore-entry object.

```
SEMAPHORE-ENTRY ::= CLASS {
    &entryID                INTEGER,
-- this value shall be unique to the semaphore object
-- that is the parent of this object
    &class                  ENUMERATED {
        simple                  (0),
        modifier                (1) },
    &semaphore              SEMAPHORE,
    &requester              ApplicationReference,
```

```
    &aaIdentifier           APPLICATION-ASSOCIATION,
    &invokeID               TRANSACTION,
-- The following field shall appear only if the semaphore is a pool semaphore
    &named-token            NAMED-TOKEN OPTIONAL,
    &priority               Priority,
    &entryState             ENUMERATED {
        queued                  (0),
        owner                   (1),
        hung                    (2) },
-- The following field shall appear only if the entryState has the value queued.
    &remainingAcqDelay      CHOICE {
        time                    Unsigned32,
        forever                 NULL } OPTIONAL,
-- The following field shall appear
-- only if the entryState has the value owner or hung.
    &remainingTimeOut       CHOICE {
        time                    Unsigned32,
        forever                 NULL } OPTIONAL,
    &abortOnTimeOut         BOOLEAN,
    &relinquishIfLost       BOOLEAN }
```

16.1.3.1 &entryID

The &entryID field identifies a semaphore-entry object and shall be unique for all the semaphore-entry objects related to a given semaphore.

16.1.3.2 &class

The &class field contains the value **modifier** if the semaphore-entry has been created by a service modified by the AttachToSemaphore Modifier, otherwise it contains the value **simple**.

16.1.3.3 &semaphore

The &semaphore field identifies the semaphore requested or owned by the semaphore-entry.

16.1.3.4 &requester

The &requester field identifies the Application Process whose request created the semaphore-entry.

16.1.3.5 &aaIdentifier

The &aaIdentifier field identifies the Application Association under which the semaphore-entry was created. This field cannot be reported through any of the MMS services.

16.1.3.6 &invokeID

The &invokeID field identifies a Transaction object in the VMD. This Transaction object exists at least as long as the semaphore-entry is waiting for the control of the semaphore if the &class field is **simple**, or as long as the semaphore has not been relinquished if the &class field is **modifier**. The possible nesting of modifiers is managed by this Transaction object.

16.1.3.7 &namedToken

The &named-token field is present if and only if the requested semaphore is a pool semaphore. It identifies the named-token object related to this semaphore.

16.1.3.8 &priority

The &priority field specifies the priority that the semaphore-entry shall have relative to other semaphore-entry objects while in the waiting list. The &priority field shall be an integer with values ranging from 0 to 127 inclusive. 0 shall represent the highest priority, 64 the normal priority and 127 the lowest priority. Treatment of priority by the MMS server is a local matter.

Priority ::= Unsigned8

normalPriority Priority ::= 64

16.1.3.9 &entryState

The &entryState field specifies the state of the semaphore-entry object. It shall have a value either **queued**, **owner**, or **hung**. The &entryState field contains the value **queued** while the semaphore-entry is in the waiting list, **owner** while it is in the owner list and the Application Association is still maintained, and **hung** while it is in the owner list and the Application Association is lost.

16.1.3.10 &remainingAcqDelay

The &remainingAcqDelay field shall specify the duration of time that a semaphore-entry can remain in the &entryState **queued**. This field only has meaning if the &entryState field has the value **queued**. The value shall be either a positive number or the value **forever**.

16.1.3.11 &remainingTimeOut

The &remainingTimeOut field shall specify the duration of time that a semaphore-entry can remain in the &entryState **owner** or **hung**. The field shall only have meaning if the &entryState field has the value **owner** or **hung**. The value is either a positive number or the value **forever**.

16.1.3.12 &abortOnTimeOut

The &abortOnTimeOut field is a boolean that specifies whether (true) or not (false) the Application Association shall be aborted if the Control Time Out occurs. The requesting MMS-user may make use of the Event Condition associated with the semaphore together with appropriate Event Actions and Event Enrollments (see clause 18) to define remedial procedures that should be performed following a Control Time Out.

NOTE There is only one Event Condition for any given semaphore. This Event Condition reflects the state of all the subordinate semaphore-entry objects in that if a Control Time Out occurs for any semaphore-entry object, this Event Condition is triggered. An MMS client may make use of the ReportSemaphoreEntryStatus service to determine which semaphore-entry caused the transition; this Entry will be in the **hung** state.

16.1.3.13 &relinquishIfLost

The &relinquishIfLost field is a boolean that specifies, if true, that the semaphore-entry shall relinquish the semaphore if the Application Association identified by the &aaIdentifier field is lost. The value false means that, in such a situation, the semaphore-entry shall not relinquish the semaphore, but rather shall place the value **hung** in the &entryState field value.

16.1.4 Model of a Semaphore-entry

The semaphore-entry model is provided in Figure 13.

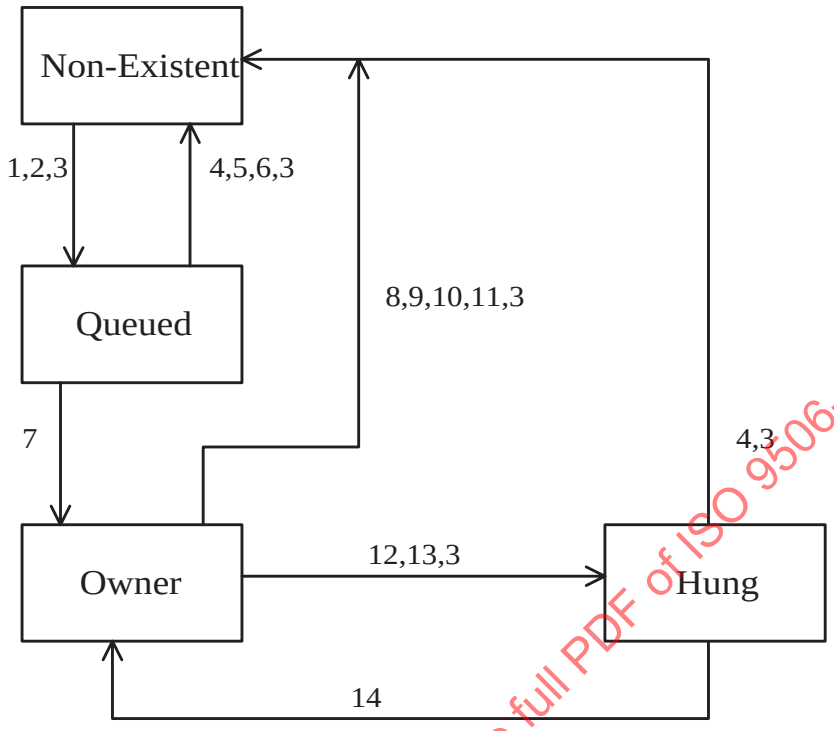

Figure 13 - Semaphore Entry model

Transitions of the model are as follows:

1	Attach To Semaphore Modifier	10	Control Time Out and Abort on Time Out and Relinquish if Connection Lost
2	Take Control request	11	Association Abort and Relinquish if Connection Lost
3	local Action	12	Association Abort and not Relinquish if Connection Lost
4	Time Out	13	Control Time Out and not Abort on Time Out
5	Cancel request	14	Preempt
6	Association Abort		
7	Semaphore Available		
8	Relinquish Control request		
9	Modified request processed		

A semaphore-entry shall be created either by a TakeControl request, or by any MMS service request modified by an AttachToSemaphore Modifier, or by a local action requesting control of the semaphore. After creation, the semaphore-entry shall be placed in a queue.

The queue shall be ordered by an algorithm dependent on the class of semaphore serving the queue. When the semaphore-entry is at the top of the queue and the semaphore is able to serve it, the semaphore-entry shall be granted control of the semaphore and shall be removed from the queue. If the semaphore-entry has been created by a TakeControl request, a Result(+) response shall be issued. If it has been created by a modified request, the modified request shall be released for further processing as specified in the service procedure under control of the Transaction object.

If the Remaining Acquisition Delay timer associated with the request expires before control has been granted, the semaphore-entry shall be deleted and a Result(-) response shall be issued.

A semaphore-entry that is in control of the semaphore shall release control of the semaphore either following a RelinquishControl request issued on the same association, or when the modified request has been processed, or by local means, depending upon the way the control had been requested. If a Control Time Out occurs, depending upon the value of the Abort on Time Out attribute,

either the association shall be aborted and the semaphore-entry processed as for an association abort, or the semaphore-entry shall be placed in the **hung** state. The related Event Condition shall be triggered.

If the application association upon which the request was received is aborted while the semaphore-entry is in control of the semaphore and the value of the &relinquishIfConnectionLost field is true, the control of the semaphore shall be released. If the application association upon which the request was received is aborted while the semaphore-entry is in control of the semaphore and the value of the &relinquishIfConnectionLost field is false, the semaphore-entry shall remain in the **hung** state until a preemptive TakeControl request is issued by any MMS-user.

Multiple requests for control of a single semaphore, issued by the same MMS client to the same MMS server are independent. Therefore the second request shall remain queued while the first request is served.

16.1.5 Model of the token semaphore

The token semaphore is described by the model in Figure 14.

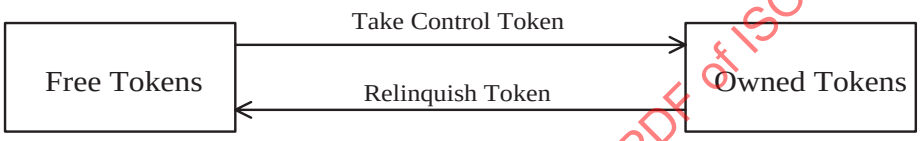

Figure 14 - Token Semaphore model

A token semaphore is modeled by a collection of identical tokens, each token evolving between the states **free** and **owned**. The total number of tokens represents the maximum number of owners of the semaphore (a token semaphore with only one token is a mutually exclusive semaphore). The state of the semaphore is defined by the number of **free** tokens and the number of **owned** tokens. The Take Control Token transition moves one token from the state **free** to the state **owned**, and the Relinquish Token transition moves one token from the state **owned** to the state **free**.

Each **owned** token is associated with a semaphore-entry in the state **owner** or **hung**. The Take Control Token transition models the association of a token with a semaphore-entry and the transition of a semaphore-entry from the state **queued** to **owner**. The Relinquish Control transition models the deletion of a semaphore-entry in the state **owner**.

At the creation of the semaphore, all the tokens shall be **free**. As soon as a semaphore-entry is created, one token shall evolve into the state **owned**. A token shall evolve from **free** to **owned** each time there is a **free** token and there is a semaphore-entry in the state **queued**, either following the release of a token or the creation of a semaphore-entry. A token shall evolve from **owned** to **free** when the associated semaphore-entry is deleted, following a RelinquishControl request or the end of processing of a AttachToSemaphore modified request, or through a local action. A preempt request shall maintain the token in the state **owned** while changing the associated semaphore-entry.

The queue of waiting semaphore-entries shall be served on a first-in-first-out basis for the entries of same priority. The algorithm used for handling prioritized queues is a local matter, and shall be specified in the Configuration and Initialization Statement (CIS) (see ISO 9506-2, clause 25).

16.1.6 Model of the Pool Semaphore

A pool semaphore may be described by the model in Figure 15.

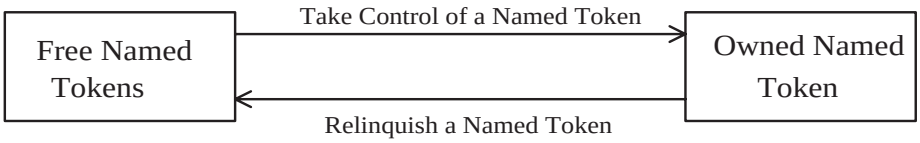

Figure 15 - Pool Semaphore model

A pool semaphore is modeled as a collection of named-tokens, each named-token evolving between the states **free** and **owned**. The difference between a token semaphore and a pool semaphore from a modelling point of view is that the tokens handled by a pool semaphore are identifiable by a name that may be specified when requesting the semaphore. With that difference, the description of the token semaphore applies to the pool semaphore.

The queue of waiting semaphore-entries shall be served on a first-in-first-out basis for the entries of same priority. If the entry at the top of the list requests a non-available named-token, this entry shall remain at the top of the list and the next entry processed. The algorithm used for handling prioritized queues is a local matter, and shall be specified in the Configuration and Initialization Statement (CIS) (see ISO 9506-2, clause 25).

16.2 TakeControl service

The TakeControl service may be used by an MMS client in order to obtain control of a semaphore.

16.2.1 Structure

The structure of the component service primitives is shown in Table 90.

Table 90 - TakeControl service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Named Token	U	U(=)			
Priority	M	M(=)			
Acceptable Delay	U	U(=)			
Control Time Out	U	U(=)			
Abort On Time Out	C	C(=)			
Relinquish if Connection Lost	M	M(=)			
Application To Preempt	U	U(=)			
Result(+)			S	S(=)	
Named Token			C	C(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.2.1.1 Argument

This parameter shall convey the parameters of the TakeControl service request.

16.2.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the &name field of the semaphore to be controlled.

16.2.1.1.2 Named Token

This optional parameter, of type Identifier, shall be provided if and only if the semaphore is a pool semaphore. If provided, it shall specify the named-token controlled by the pool semaphore over which the MMS client wishes to be granted control. If this parameter is not provided and if the semaphore is a pool semaphore, the choice of the allocated named-token shall be made by the MMS server.

16.2.1.1.3 Priority

This parameter, of type integer, shall identify the priority that this TakeControl service request should have as compared to other (possible) TakeControl requests, as well as to services using the AttachToSemaphore Modifier, that are awaiting the semaphore identified in the TakeControl request.

16.2.1.1.4 Acceptable Delay

This parameter shall indicate either the duration of time in milliseconds for which the requesting user is willing to wait for control to be allocated, or the value **forever**. If a zero value is specified for this parameter, this means that no delay is acceptable. (That is, if the semaphore or named-token is not immediately available, the service shall fail.) The granularity of one millisecond is not required. The granularity supported by the MMS server shall be specified in the Configuration and Initialization Statement (CIS) (see ISO 9506-2, clause 25).

16.2.1.1.5 Control Time Out

This parameter shall specify either the duration of time in milliseconds for which control of the semaphore may be held (after it is obtained), or the value **forever**. If this time limit is exceeded, and the value of the Abort On Time Out parameter is true, the Application Association shall be aborted with a provider abort and the semaphore-entry shall be relinquished or changed to the **hung** state according to the value of the Relinquish If Connection Lost parameter. If this time limit is exceeded and the value of the Abort On Time Out parameter is false, the semaphore-entry shall be changed to the **hung** state. In either case, the related Event Condition shall be triggered. The granularity of one millisecond is not required. The granularity supported by the MMS server shall be specified in Configuration and Initialization Statement (CIS) (see ISO 9506-2, clause 25).

16.2.1.1.6 Abort On Time Out

This parameter, of type boolean, shall be provided if the Control Time Out parameter is provided. The value true shall specify that the Association shall be aborted in case the Control Time Out expires. The value false shall specify that the related Event Condition shall be signalled.

16.2.1.1.7 Relinquish If Connection Lost

This parameter, of type boolean, shall specify if true that the MMS server shall relinquish the semaphore if the owner of the semaphore loses the ability to control it, either due to the loss of the association used for acquiring the semaphore, or to a local failure. The value false shall mean that in the same situation the MMS server shall maintain the semaphore in the **owned** state, with the associated semaphore-entry in the **hung** state.

16.2.1.1.8 Application To Preempt

This optional parameter, of type Application Reference, shall specify the owner of a semaphore-entry that is in the **hung** state. The presence of this parameter shall indicate that the requester of the service wants to take control of a semaphore by preempting control of a semaphore-entry having state **hung** whose owner matches the Application To Preempt parameter.

16.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

16.2.1.2.1 Named Token

If the specified semaphore is a pool semaphore, this parameter shall either specify the named-token specified in the Named Token parameter of the request, or a named-token allocated from the pool of named-tokens if the Named Token parameter was not provided in the request. If the specified semaphore is a token semaphore, this parameter shall not be present.

16.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.2.2 Service Procedure

16.2.2.1 Actions if Application To Preempt parameter is not present

16.2.2.1.1 Preconditions

The MMS server shall:

a) verify that the semaphore identified by the Semaphore Name parameter exists.

b) if the Named Token parameter is present in the indication primitive, verify that the semaphore is a pool semaphore and that a named-token corresponding to this parameter exists.

c) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

d) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the semaphore are satisfied for the service class = LOAD.

If any of these conditions is not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED. The remainder of this procedure shall be skipped.

16.2.2.1.2 Step 1

The MMS server shall create a semaphore-entry object and add it to the &Requesters field of the semaphore. The fields of the semaphore-entry shall be initialized as follows:

a) The &entryID field shall contain a locally selected value unique to this semaphore.

b) The &class field shall be set to **simple**.

c) The &semaphore field shall be initialized to the semaphore indicated by the Semaphore Name parameter.

d) The &requester field shall identify the Application Process that issued the request.

e) The &aaIdentifier field shall be initialized to a value identifying the application association over which the request was received.

f) The &invokeID field shall be initialized to the value of the Invoke ID parameter of the indication (see 5.5).

g) If the semaphore is a pool semaphore, the &namedToken field shall be initialized to the value of the Named Token parameter of the indication if provided, otherwise it shall be empty.

h) The &priority field shall be initialized to the value of the Priority parameter.

i) The &entryState shall be initialized to the value **queued**.

j) If no value of the Acceptable Delay parameter is provided, the value of the &remainingAcqDelay field shall be set to **forever**. If a value of this parameter is provided, the &remainingAcqDelay field shall be initialized to the value of the Acceptable Delay parameter, and the associated Acquisition Delay Timer activated.

k) If no Control Time Out parameter is provided, the value of the &remainingTimeOut field shall be set to **forever**. If a value of this parameter is provided, the &remainingTimeOut field shall be initialized to the value of the Control Time Out parameter.

l) If the Abort On Time Out parameter is present, the &abortOnTimeOut field shall be initialized to the value of the Abort On Time Out parameter.

m) The &relinquishIfLost field shall be initialized to the value of the Relinquish If Connection Lost parameter.

16.2.2.1.3 Step 2 - Pool semaphore

If the semaphore is a pool semaphore and if the Named Token parameter is present in the indication primitive, wait until the named-token identified by this parameter is in the **free** state. If the Named Token parameter was not present in the indication primitive, wait until there is any named-token in the **free** state and place its &name field in the &namedToken field of the semaphore-entry. Then:

a) Change the &state field of this named-token to **owned**.

b) Remove the semaphore-entry from the &Requesters field of the semaphore.

c) Place the semaphore-entry in the &Owners field of the semaphore.

16.2.2.1.4 Step 2 - Token semaphore

If the semaphore is a token semaphore, wait until the &numberOfOwnedTokens field is less than the &numberOfTokens field of the semaphore. Then:

a) Increase by one the &numberOfOwnedTokens field of the semaphore.

b) Remove the semaphore-entry from the &Requesters field of the semaphore.

c) Place the semaphore-entry in the &Owners field of the semaphore.

16.2.2.1.5 Step 3

Return a Result(+).

16.2.2.2 Actions if Application To Preempt parameter is present

16.2.2.2.1 Preconditions

The MMS server shall:

a) verify that the semaphore identified by the Semaphore Name parameter exists.

b) verify that there exists at least one semaphore-entry whose &requester field matches the Application to Preempt parameter and whose &entryState field is **hung**.

c) if the semaphore is a pool semaphore and the Named Token parameter is present in the indication primitive, verify that the Named Token parameter matches the &namedToken field of the semaphore-entry.

d) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

e) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the semaphore are satisfied for the service class = LOAD.

If any of these conditions is not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED. The remainder of this procedure shall be skipped.

16.2.2.2.2 Step 1

If there is more than one semaphore-entry that matches the Preconditions, the MMS server shall choose one of them (local matter) to preempt. For the selected semaphore-entry, the MMS server shall:

a) replace the current value of the &requester field with the Application Reference value of the MMS client.

b) replace the current value of the &aaIdentifier field with a value identifying the Application Association over which this request was received.

c) replace the &relinquishIfLost field with the value of the Relinquish If Connection Lost parameter of the indication.

d) if the Remaining Control Time Out parameter is not present in the indication, set the value of the &remainingTimeOut field to **forever** and deactivate any associated Remaining Control Time Out Timer.

e) if a Remaining Control Time Out parameter is present in the indication, set the &remainingTimeOut field to the value in the indication and activate the associated timer.

f) set the &entryState field to **owner**.

g) set the &class field to **simple**.

Return a Result(+).

16.3 RelinquishControl service

The RelinquishControl service may be used by an MMS-user to relinquish control of a semaphore for which control is held.

16.3.1 Structure

The structure of the component service primitives is shown in Table 91.

Table 91 - RelinquishControl service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Named Token	C	C(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.3.1.1 Argument

This parameter shall convey the parameters of the RelinquishControl service request.

16.3.1.1.1 Semaphore Name

This parameter, of type Object Name, shall be the name of the semaphore for which control is to be relinquished.

16.3.1.1.2 Named Token

This parameter, of type Identifier, shall be provided if the semaphore is a pool semaphore and shall not be provided otherwise. It shall specify the named-token to be relinquished and shall be the same as the named-token from the Result(+) parameter of the TakeControl.Confirm service primitive in which control was granted, if issued.

16.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

16.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.3.2 Service Procedure

16.3.2.1 Preconditions

The MMS server shall:

a) verify that the semaphore identified by the Semaphore Name exists.

b) verify that there is at least one semaphore-entry on the &Owners field of the semaphore whose &requester field references the Application Process of the MMS client.

c) if the Named Token parameter is present in the indication, that the &namedToken field of this semaphore-entry references this Named Token, and that the Named Token is in the **owned** state .

If any of these conditions is not satisfied, return a Result(-) and skip the remainder of this procedure.

16.3.2.2 Actions

If there are multiple semaphore-entries that meet the criteria of this service procedure, the MMS server shall select one entry (local matter) for deletion.

For the selected semaphore-entry. the MMS server shall:

a) if the semaphore is a pool semaphore, change the state of the Named Token indicated by the &namedToken field of the semaphore-entry to **free**.

b) if the semaphore is a token semaphore, decrease by one the value of the &numberOfOwnedTokens field of the semaphore.

c) remove the semaphore-entry from the &Owners field of the semaphore.

d) delete the semaphore-entry.

Return a Result(+).

16.4 DefineSemaphore service

The DefineSemaphore service may be used by an MMS client to create a token semaphore at the MMS server.

16.4.1 Structure

The structure of the component service primitives is shown in Table 92.

Table 92 - DefineSemaphore service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Number of Tokens	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.4.1.1 Argument

This parameter shall convey the parameters of the DefineSemaphore service request.

16.4.1.1.1 Semaphore Name

This parameter, of type Object Name, shall be the name to be associated with the semaphore to be created. It shall be a VMD-specific name.

16.4.1.1.2 Number of Tokens

This parameter, of type integer, shall specify the number of owners allowed to control the semaphore simultaneously.

16.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

16.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.4.2 Service Procedure

16.4.2.1 Preconditions

The MMS server shall:

a) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

b) verify that no semaphore already exists whose &name field is equal to the Semaphore Name parameter.

If any of these conditions is not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED. The remainder of this procedure shall be skipped.

16.4.2.2 Actions

The MMS server shall create the semaphore and initialize it as follows:

a) The &name field shall be set to the value of the Semaphore Name parameter.

b) The &accessControl field shall be set to reference an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) The new semaphore shall be added to the &Semaphores field of the Access Control List object referenced by the &accessControl field of this semaphore.

d) The &class field shall be set to **token**.

e) The &numberOfTokens field shall be set to the value of the Number Of Tokens parameter.

f) The &numberOfOwnedTokens field shall be initialized to the value zero (0).

g) The &Owners field shall be empty

h) The &Requesters field shall be empty.

i) An Event Condition object shall be created. The Event Condition object shall be initialized as follows:

   1) The &name field shall be set equal to the value of the &name field of this semaphore.

   2) The &accessControl field shall be initialized to reference an Access Control List object that will report the value of MMS Deletable as false (see 9.1.4). The predefined symbol 'M_NonDeletable' (see 25.3.2.2) may be used for this purpose.

3) This Event Condition shall be added to the &EventConditions field of the Access Control List field referenced by the &accessControl field of this Event Condition.

4) The &ecClass field shall be set to **network-triggered**.

5) The &ecState field shall be set to **disabled**.

6) The &priority field shall be set to **normalPriority**.

7) The &severity field shall be set to **normalSeverity**.

8) The &EventEnrollments field shall be set to empty.

j) The &eventCondition field of the semaphore shall be set to reference this Event Condition object.

Return a Result(+).

16.5 DeleteSemaphore service

The DeleteSemaphore service may be used by an MMS-user in order to delete a semaphore if such deletion is permitted.

16.5.1 Structure

The structure of the component service primitives is shown in Table 93.

Table 93 - DeleteSemaphore service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.5.1.1 Argument

This parameter shall convey the parameter of the DeleteSemaphore service request.

16.5.1.1.1 Semaphore Name

This parameter, of type Object Name, shall be the name of the semaphore to be deleted.

16.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

16.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.5.2 Service Procedure

16.5.2.1 Preconditions

The MMS server shall:

a) verify that the semaphore exists.

b) verify that the &Owners field is empty.

c) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE.

d) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the semaphore are satisfied for the service class = DELETE.

If any of these conditions is not satisfied, the service request fails and a Result(-) shall be returned.

16.5.2.2 Actions

The MMS server shall:

a) remove the reference to this semaphore from the &Semaphores field of the Access Control List object referenced by the &accessControl field of the semaphore.

b) remove the reference to the Event Condition object referenced by the &eventCondition field of the semaphore from the &EventConditions field of the Access Control List object referenced by the &accessControl field of this Event Condition.

c) delete the specified semaphore and the Event Condition object referenced by the &eventCondition field of the semaphore.

Return a Result(+).

16.6 ReportSemaphoreStatus service

The ReportSemaphoreStatus service may be used by an MMS client to obtain the status of a semaphore.

16.6.1 Structure

The structure of the component service primitives is shown in Table 94.

Table 94 - ReportSemaphoreStatus service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Class			M	M(=)	
Number Of Tokens			M	M(=)	
Number Of Owned Tokens			M	M(=)	
Number Of Hung Tokens			M	M(=)	
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.6.1.1 Argument

This parameter shall convey the parameters of the ReportSemaphoreStatus service request.

16.6.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the token or pool semaphore for which the status is to be supplied.

16.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

16.6.1.2.1 MMS Deletable

This parameter, of type boolean, shall specify if true that the semaphore may be deleted using the DeleteSemaphore service. Subclause 9.1.4 specifies the value to be returned by this parameter.

16.6.1.2.2 Class

This parameter, of type integer, shall specify the class of the semaphore and shall have either the value **token**, or **pool**.

16.6.1.2.3 Number of Tokens

This parameter, of type integer, shall specify the maximum number of owners allowed by the semaphore.

16.6.1.2.4 Number Of Owned Tokens

This parameter, of type integer, shall specify the current number of owned tokens of the semaphore, whose associated semaphore-entry is not in the **hung** state.

16.6.1.2.5 Number Of Hung Tokens

This parameter, of type integer, shall specify the number of owned tokens whose associated semaphore-entry is in the **hung** state.

16.6.1.2.6 Access Control List

This parameter, of type Identifier, shall indicate the name of the Access Control List object that controls access to this semaphore. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

16.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.6.2 Service Procedure

16.6.2.1 Preconditions

The MMS server shall verify that the semaphore identified by the Semaphore Name parameter exists. If this condition is not satisfied, a Result(-) shall be returned.

16.6.2.2 Actions

The MMS server shall return the response service primitive containing values of the parameters corresponding to this semaphore.

16.7 ReportPoolSemaphoreStatus service

The ReportPoolSemaphoreStatus service may be used by a MMS client to obtain the name and the state of the named-tokens controlled by a pool semaphore at the MMS server.

16.7.1 Structure

The structure of the component service primitives is shown in Table 95.

Table 95 - ReportPoolSemaphoreStatus service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
Name To Start After	U	U(=)			
Result(+)			S	S(=)	
List Of Named Token			M	M(=)	
Free Named Token			S	S(=)	
Owned Named Token			S	S(=)	
Hung Named Token			S	S(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.7.1.1 Argument

This parameter shall convey the parameters of the ReportPoolSemaphoreStatus service request.

16.7.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the semaphore to be reported.

16.7.1.1.2 Name To Start After

This optional parameter shall indicate, if present, that the MMS client requests only the sublist beginning with a name other than the first name in the list. If the Name To Start After parameter does not match an existing named-token at the MMS server, the sublist shall begin with the first name to follow the Name To Start After parameter.

16.7.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

16.7.1.2.1 List Of named-token

This parameter shall be a list, possibly empty, ordered by the &name field of the named-token according to the collation sequence defined in 5.4.2. Each element of the list shall be one of the following parameters:

16.7.1.2.1.1 Free named-token

This parameter, of type character string, shall contain the name of a named-token whose &state field has the value **free**.

16.7.1.2.1.2 Owned named-token

This parameter, of type character string, shall contain the name of a named-token whose &state field has the value **owned**, and whose associated semaphore-entry is not in the **hung** state.

16.7.1.2.1.3 Hung named-token

This parameter, of type character string, shall contain the name of a named-token whose &state field has the value **owned**, and whose associated semaphore-entry is in the **hung** state.

16.7.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional ReportPoolSemaphoreStatus requests are necessary to retrieve all the requested information. If true, more requests are necessary. If false, then either the List Of Named Token parameter contains the last item in the list, or the List Of Named Token parameter is empty.

16.7.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.7.2 Service Procedure

16.7.2.1 Preconditions

The MMS server shall verify that:

a) a semaphore whose &name field matches the Semaphore Name parameter exists;

b) the semaphore is a pool semaphore.

If any of these conditions is not met, the MMS server shall return a Result(-).

16.7.2.2 Actions

The MMS server shall prepare a list of named-tokens from the &NamedTokens field of the pool semaphore, ordered by the collation sequence defined in 5.4.2. The MMS server shall return a List Of Named Token parameter, derived from this complete list. The List Of Named Token parameter shall begin at the beginning of the list if the Name To Start After parameter is not provided in the service indication; otherwise, it begins at the first name in the list after the value provided by the Name To Start After parameter.

The More Follows parameter shall be set to true if more items remain to be reported after this list is processed; otherwise this parameter shall be set to false. If the List of named-tokens is empty, the More Follows parameter shall be false.

The number of items to be returned in this list (i.e. the size of the subset of the complete list) is a local matter.

16.8 ReportSemaphoreEntryStatus service

The ReportSemaphoreEntryStatus service may be used by an MMS client to obtain the status of the semaphore-entries dependent on a semaphore at the MMS server.

16.8.1 Structure

The structure of the component service primitives is shown in Table 96.

Table 96 - ReportSemaphoreEntryStatus service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Semaphore Name	M	M(=)			
State	M	M(=)			
Entry ID To Start After	U	U(=)			
Result(+)			S	S(=)	
List Of Semaphore Entry			M	M(=)	
Entry ID			M	M(=)	
Entry Class			M	M(=)	
Application Reference			M	M(=)	
Named Token			C	M(=)	
Priority			M	M(=)	
Remaining Time Out			M	M(=)	
Abort On Time Out			M	M(=)	
Relinquish if Connection Lost			M	M(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

16.8.1.1 Argument

This parameter shall convey the parameters of the ReportSemaphoreEntryStatus service request.

16.8.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the name of the semaphore for which the status of its semaphore-entries is to be supplied.

16.8.1.1.2 State

This parameter shall specify the &entryState field of the semaphore-entry. This parameter may have the value QUEUED, OWNER, or HUNG.

16.8.1.1.3 Entry ID To Start After

This optional parameter, of type octetstring, shall indicate, if present, that the MMS client requests only the sublist of semaphore-entries beginning after the semaphore-entry whose Entry ID is provided by this parameter to be returned. If the Entry ID does not match an existing semaphore-entry at the MMS server, the sublist shall begin with the first semaphore-entry whose &entryID field is numerically greater than the Entry ID parameter.

16.8.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

16.8.1.2.1 List Of Semaphore-entry

This parameter shall be a list, possibly empty, ordered by the &entryID field of the semaphore-entry. Each element of the list shall contain the following parameters:

16.8.1.2.1.1 Entry ID

This parameter, of type octetstring, shall be the &entryID field of a semaphore-entry.

16.8.1.2.1.2 Entry Class

This parameter shall contain the &class field of the semaphore-entry. It may have the value **simple** or **modifier**.

16.8.1.2.1.3 Application Reference

This parameter, of type ApplicationReference, shall identify the MMS-user that created the semaphore-entry.

16.8.1.2.1.4 Named Token

This optional parameter, of type Identifier, shall be present if the semaphore is a pool semaphore. Otherwise it shall not be present. It shall contain the value of the &namedToken field of the semaphore-entry.

16.8.1.2.1.5 Priority

This parameter, of type integer, shall contain the value of the &priority field of the semaphore-entry.

16.8.1.2.1.6 Remaining Time Out

This optional parameter, of type integer, shall contain either the &remainingAcqDelay field if the semaphore-entry is in the state **queued**, or the &remainingTimeOut field if the semaphore-entry is in the state **owner**.

16.8.1.2.1.7 Abort On Time Out

This parameter, of type boolean, shall contain the value of the &abortOnTimeOut field of the semaphore-entry.

16.8.1.2.1.8 Relinquish if Connection Lost

This parameter, of type boolean, shall contain the value of the &relinquishIfLost field of the semaphore-entry.

16.8.1.2.2 More Follows

This parameter, of type boolean, shall indicate whether additional ReportSemaphoreEntryStatus requests are necessary to retrieve the information for all semaphore-entries. If true, more requests are necessary. If false, either the List Of Semaphore Entry parameter contains the last item in the list, or it is empty.

16.8.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

16.8.2 Service Procedure

16.8.2.1 Preconditions

The MMS server shall verify that the semaphore identified by the Semaphore Name parameter exists. If this condition is not satisfied, a Result(-) shall be returned.

16.8.2.2 Actions

The MMS server shall prepare a list of semaphore-entries as follows:

a) if the State parameter of the service request is QUEUED, the list shall be the &Requesters field of the semaphore;

b) if the State parameter of the service request is OWNER, the list shall be those semaphore-entries from the &Owners field of the semaphore whose &entryState field is **owner**;

c) if the State parameter of the service request is HUNG, the list shall be those semaphore-entries from the &Owners field of the semaphore whose &entryState field is **hung**.

The list shall be ordered by the numerical value of the &entryID field of the semaphore-entry. The List Of Semaphore Entry parameter shall be constructed from this list, either in its entirety, or in a sublist drawn from this list.

If the Entry ID To Start After parameter is present in the service indication, the sublist shall begin with the first semaphore-entry whose &entryID field is numerically greater than the value of the Entry ID To Start After parameter. If the Entry ID To Start After parameter is not present in the service indication, the list (or sublist) shall begin with the first semaphore-entry on the list.

If the list to be returned is too large to be sent in a single service response, the MMS server shall return a sublist of the total list and shall return the More Follows parameter with the value true. The MMS client may send another service request, with the Entry ID To Start After parameter containing the last Entry ID returned.

NOTE Due to the dynamics of semaphore-entry lists, the range and the state of a specific item of the list may change between two service requests and the application should be aware of this. This service is designed for handling recovery procedures in the case of deadlocks or connection aborts, and such situations are static.

A Result(+) shall be returned containing the List of Semaphore Entry parameter.

16.9 AttachToSemaphore Modifier

The AttachToSemaphore Modifier is provided so that the processing of a service request may be delayed at a MMS server until the control of a semaphore has been granted by this service. Services that are modified are not acted on immediately, but are placed in a queue corresponding to a semaphore at the MMS server; when this service request rises to the top of the queue, it is acted on.

16.9.1 Structure

The structure of the component service primitives is shown in Table 97.

Table 97 - AttachToSemaphore Modifier

Parameter Name	Req	Ind	CBB
Attach To Semaphore	S	S(=)	
Semaphore Name	M	M(=)	
Named Token	C	C(=)	
Priority	M	M(=)	
Acceptable Delay	U	U(=)	
Control Time Out	U	U(=)	
Abort On Time Out	C	C(=)	
Relinquish if Connection Lost	M	M(=)	

16.9.1.1 Attach To Semaphore

The Attach To Semaphore parameter is provided as a parameter of the List of Modifier parameter for each confirmed service request (see 5.6). The sub-parameters of the Attach To Semaphore modifier are specified as follows:

16.9.1.1.1 Semaphore Name

This parameter, of type Object Name, shall specify the &name field of the semaphore under which the modified service request is to be controlled.

16.9.1.1.2 Named Token

This parameter, of type character string, shall be provided if the semaphore is a pool semaphore and shall not be provided otherwise. It shall specify the named-token controlled by the pool semaphore the MMS client wishes to use for control. The dynamic allocation of a named-token by the MMS server is not allowed, since the modified request could not identify the allocated named-token.

16.9.1.1.3 Priority

This parameter, of type integer, shall identify the priority that this modified service request should have as compared to (possible) other TakeControl requests, as well as to services using the AttachToSemaphore Modifier waiting for the same semaphore.

16.9.1.1.4 Acceptable Delay

This optional parameter, of type integer, shall indicate the duration of time for which the MMS client is willing to wait for control to be allocated. If a zero value is specified for acceptable delay, this means that no delay is acceptable. (That is, if the semaphore or named-token is not immediately available, the service to which the Modifier is attached shall fail.) If no value is specified for acceptable delay, this shall be interpreted as meaning that any delay is acceptable (that is "wait forever"). The granularity of one millisecond is not required. The granularity supported by the MMS server shall be specified in Configuration and Initialization Statement (CIS) (see ISO 9506-2, clause 25).

16.9.1.1.5 Control Time Out

This optional parameter, of type integer, shall specify the duration of time in milliseconds for which control of the semaphore may be held (after it is obtained). If no value is provided, no control time out applies, and the semaphore may be held indefinitely. The granularity of one millisecond is not required. The granularity supported by the MMS server shall be specified in Configuration and Initialization Statement (CIS) (see ISO 9506-2, clause 25).

16.9.1.1.6 Abort On Time Out

This boolean parameter shall be provided if the Control Time Out parameter is provided. The value true shall mean that the Application Association shall be aborted if the Control Time Out expires. The value false shall mean that the Application Association is to be maintained and the related Event Condition signalled if the Control Time Out expires.

16.9.1.1.7 Relinquish If Connection Lost

This boolean parameter shall specify if true that the MMS server shall relinquish the semaphore if the owner of the semaphore loses the ability to control it, either due to the loss of the association used for acquiring the semaphore, or to a local failure. The value false shall mean that in the same situation the MMS server shall place the semaphore-entry in the **hung** state.

16.9.2 Service Procedure

16.9.2.1 Preconditions

The MMS server shall verify that:

a) all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD;

b) all the conditions in the Access Control List object referenced by the &accessControl field of the semaphore are satisfied for Service Class = LOAD.

c) the semaphore identified by the Semaphore Name parameter exists.

d) the named-token specified in the service request (if present) exists.

e) the delay specified by the Acceptable Delay parameter has not expired before control of the semaphore is effective.

f) the service request has not been cancelled by a Cancel service request.

If any of these conditions is not satisfied, a Result(-) shall be returned for the modified service with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED. The remainder of this procedure shall be skipped.

16.9.2.2 Actions

When the MMS server processes this modifier as part of the transaction object processing of the modified service (see 7.4.3), it shall create a semaphore-entry object and add it to the &Requesters field of the semaphore. The fields of the created semaphore-entry shall be initialized as follows:

a) the &entryID field shall contain a locally defined value that shall be unique for each semaphore.

b) the &class field shall be set to **modifier**.

c) the &semaphore field shall be set to the semaphore indicated by the Semaphore Name parameter.

d) the &requester shall be initialized to the value identifying the Application Process issuing the service request.

e) the &aaIdentifier attribute shall be initialized to a value identifying the application association on which the service indication was received.

f) the &invokeID field shall be set to the value of the Invoke ID parameter of the indication. This field references the Transaction object that controls the execution of this service request.

g) if the semaphore is a pool semaphore, the &namedToken field shall be initialized to the value of the Named Token parameter of the service indication.

h) the &priority field shall be initialized to the value of the Priority parameter.

i) if no value of the Acceptable Delay parameter is provided, the value of the &remainingAcqDelay field shall be set to **forever**. If a value is provided, the &remainingAcqDelay field shall be initialized to the value of the Acceptable Delay parameter and the associated timer activated.

j) if no value of the Control Time Out parameter is provided, the value of the &remainingTimeOut field shall be set to **forever**. If a value is provided, the &remainingTimeOut field shall be initialized to the value of the Control Time Out parameter.

k) the &abortOnTimeOut field shall be initialized to the value of the Abort On Time Out parameter.

l) the &relinquishIfLost field shall be initialized to the value of the Relinquish If Connection Lost parameter.

m) the &entryState shall be initialized to the value **queued**.

When control of the semaphore is granted, the service request shall be released for further processing, under control of the Transaction object. Upon completion of all processing associated with the modified service, the semaphore shall be relinquished. The semaphore shall also be relinquished (1) upon any abnormal completion of the request such as a Cancel service, or (2) by an abort of the association if the &relinquishIfLost field so specifies.

16.10 Conformance

The Semaphore Management Services define parameter conformance requirements for the VMD. These requirements are described below.

16.10.1 Support for Time

The Configuration and Initialization Statement (CIS) shall provide the granularity of time supported in the processing of the &remainingAcqDelay and &remainingTimeOut fields of a semaphore-entry object.

16.10.2 Support for Priority

The Configuration and Initialization Statement (CIS) shall specify the algorithm executed in the processing of the &priority field of a semaphore-entry object.

17 Operator Communication services

This clause provides an object model for the following object:

OPERATOR-STATION

This clause specifies the following services:

Input
Output

The Operator Communication Services provide a mechanism for communicating with an Operator Station that allows display of data, entry of data, or both.

NOTE The MMS Operator Communication Services are intended to be restrictive and simple. In the OSI environment, general virtual terminal capabilities, including extensive display management, are provided through the ISO Virtual Terminal (VT) service and protocol, defined in ISO 9040 and ISO 9041, respectively. Systems requiring a general operator communication facility should make use of VT services instead of the MMS Operator Communication Services. Similar services are available in other network environments.

17.1 The Operator Communications Model

The Operator Communication Services define an Operator Station object to describe how the Input and Output services are used. This International Standard does not specify a flow control mechanism to manage the input and output function for these objects. It is the responsibility of implementation to provide locally defined flow control mechanisms to ensure data integrity on the Operator Station for multiple transactions for a single MMS-user.

NOTE When more than one MMS-user is competing for the operator station object, the MMS-user should use the Access Control facility to manage the control of the operator station object. The MMS Server may require the use of a semaphore by specifying it as a condition in an Access Control List for accepting input and output to the Operator Station.

17.1.1 The Operator Station object

This clause introduces the model of the Operator Station object.

```
OPERATOR-STATION ::= CLASS {
    &name                   Identifier,
-- shall be unique within its range of specification (VMD)
    &accessControl          ACCESS-CONTROL-LIST,
    &stationType            ENUMERATED {
        entry                   (0),
        display                 (1),
        entry-display           (2) },
-- The following field shall appear if stationType is entry or entry-display
    &inputBuffer            MMSString OPTIONAL,
-- The following field shall appear if stationType is display or entry-display
    &outputBuffers          SEQUENCE OF MMSString OPTIONAL,
    &state                  ENUMERATED {
        idle                    (0),
        display-prompt-data     (1),
        waiting-for-input       (2),
        input-buffer-filled     (3),
        output-buffers-filled   (4) } }
```

17.1.1.1 &name

The &name field uniquely identifies the Operator Station object.

17.1.1.2 &accessControl

The &accessControl field is a reference to an Access Control List object that provides conditions under which this Operator Station may be read, written, or have its access control changed.

17.1.1.3 &stationType

The &stationType field indicates the type of station for the Operator Station object. There are three possible values for this field: **entry**, **display**, and **entry-display**.

17.1.1.4 &inputBuffer

The &inputBuffer field contains the value of the Input String parameter of the Input service response. Operator Station objects whose &stationType field is **entry** or **entry-display** have this field.

17.1.1.5 &outputBuffers

The &outputBuffers field specifies zero or more output buffers. These buffers are used either for the List Of Output Data parameter for the Output service request, or for the List Of Prompt Data parameter for the Input service request. Operator Station objects whose &stationType field is **display** or **entry-display** have this field.

17.1.1.6 &state

The &state field contains the state of the Operator Station object. An Operator Station object of type **entry** may be in one of three states: **idle**, **waiting-for-input-string**, and **input-buffer-filled**. An Operator Station object of type **display** may be in one of two states: **idle** and **output-buffers-filled**. An Operator Station object of type **entry-display** may be in one of five states: **idle**, **display-list-of-prompt-data**, **waiting-for-input-string**, **input-buffer-filled**, and **output-buffers-filled**. In the **idle** state, the object may accept Input or Output service requests, depending on its type. In any other state, the operator station object is busy executing a service request and may be unavailable for additional service requests depending upon the local flow control mechanism used by the implementation.

17.1.2 Relationship between the MMS object and the Physical Device

The relationship between the physical operator station device and the &inputBuffer field and the &OutputBuffers field of the operator station object is specified by a pair of abstract functions that work with physical data storage areas. The &inputBuffer field of the MMS operator station object is associated with an input buffer, which is a physical storage area containing the value of the Input String entered by the operator. The &OutputBuffers field of the MMS operator station object is associated with a set of output buffers, which is a set of physical storage areas containing the values of either the List Of Prompt Data parameter from the Input service request, or the List Of Output Data parameter from the Output service request. The abstract functions that provide the mapping of the values contained in these buffers to the physical device in a way that allows the operator to interact with the Input and Output service requests are described below.

17.1.2.1 The D-Put Function

The D-Put function obtains the values contained in the set of output buffers and writes these values to the physical display of the operator station device. This occurs when the List Of Output Data parameter values need to be displayed for the Output service request, or when the List Of Prompt Data parameter values need to be displayed for the Input service request.

17.1.2.2 The E-Get Function

The E-Get function obtains the Input String parameter value that is entered by the operator and writes it to the input buffer. This function completes its process when a locally defined end-of-input indication is received.

17.1.3 Operator Station State Diagram

The state diagram shown in Figure 16 shows the state transitions possible for an Operator Station object of type **entry-display**. If the Operator Station object is of type **entry**, state transitions 2 and 3 shall not be permitted. If the Operator Station object is of type **display**, the state transitions 1, 4, 5, and 6 shall not be permitted.

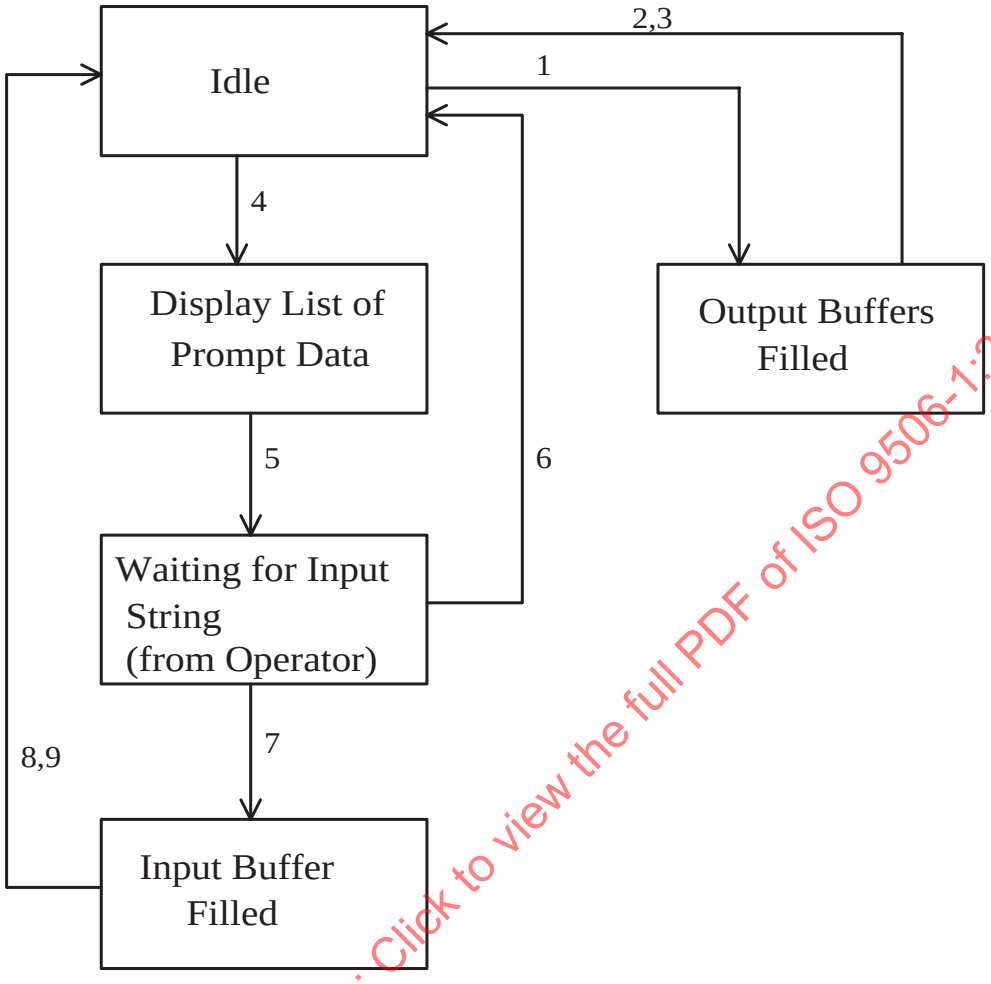

Figure 16 - Operator Station State Diagram

Transitions of the Operator Station state diagram are as follows:

1 - Output.indication
2 - Output.response (+)
3 - Output.response (-)
4 - Input.indication
5 - (D-Put function finished displaying List Of Prompt Data, if any. If present, Input Time Out begins.)
6 - Input.response (-) due to a time out
7 - (E-Get function finished entering the Input String into input buffer. If present, Input Time Out stops.)
8 - Input.response (+)
9 - Input.response (-)

17.2 Input service

The Input service may be used by an MMS client to request data from an Operator Station object of type **entry** or **entry-display** at the MMS server.

17.2.1 Structure

The structure of the component service primitives is shown in Table 98.

Table 98 - Input service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Operator Station Name	M	M(=)			
Echo	M	M(=)			
List of Prompt Data	U	U(=)			output
Input Time Out	U	U(=)			
Result(+)			S	S(=)	
Input String			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

17.2.1.1 Argument

This parameter shall convey the service specific parameters of the Input service request.

17.2.1.1.1 Operator Station Name

This parameter, of type Identifier, shall identify the Operator Station object from which the Input String is requested.

17.2.1.1.2 Echo

This parameter, of type boolean, shall indicate whether (true) or not (false) the value of the Input String parameter is to be displayed on the Operator Station.

17.2.1.1.3 List Of Prompt Data

This optional parameter shall be a list of character strings. This parameter shall only be present if the Operator Station object type is **entry-display** and the **output** service CBB is supported.

17.2.1.1.4 Input Time Out

This optional parameter, of type integer, shall specify a time out period (in seconds) for an operator to complete the process of entering the value of the Input String on the Operator Station.

17.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

17.2.1.2.1 Input String

This parameter, of type character string, is the data provided by the operator. The Input String parameter in the Input service response shall contain the value of one input buffer. The determination of the end of a line of Input String is a local matter, but the end of line indication shall not be included in the Input String.

NOTE Multiple line inputs require multiple Input service requests.

17.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

17.2.2 Service Procedure

17.2.2.1 Preconditions

The MMS server shall verify that:

a) an Operator Station object corresponding to the Operator Station Name parameter exists;

b) the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = WRITE;

c) the conditions in the Access Control List object referenced by the &accessControl field of the Operator Station object are satisfied for the service class = WRITE;

d) the Operator Station object is in the **idle** state;

e) the Operator Station object can support the Input Time Out parameter specified in the Input service request;

f) if the List Of Prompt Data parameter is present, that the Operator Station object has a &stationType field equal to **entry-display** and the **output** service CBB is supported;

g) if the Echo parameter is true, that the Operator Station object has a &stationType field equal to **entry-display**;

h) a time out has not occurred.

If any of these conditions is not met, a Result(-) shall be returned and the remainder of this procedure shall be skipped.

17.2.2.2 Actions

If the List of Prompt Data parameter is present, this parameter shall be copied into the &outputBuffers field of the Operator Station object. The D-Put function shall display the value of the &OutputBuffers on the display. Each element of the list shall represent a single line on the display.

If the Input Time Out parameter is present in the service indication, the MMS server shall begin timing this Input service request when the processing of the transaction object has begun, or, if the List Of Prompt Data parameter is present, immediately after the D-Put function has finished writing the value of this parameter to the display of the Operator Station. Timing shall stop for this Input service request immediately after the E-Get function has written the Input String entered by the operator to the input buffer.

The values for the Input Time Out parameter ranges from 0 to 2**31-1. If this parameter is present and the specified time out period expires, an error result shall be returned. If this parameter is absent, an unlimited time out period is indicated. A zero value for this parameter shall result in an Result(-) unless the E-Get function indicates that it has already written the Input String entered by the operator into the &inputBuffer.

The MMS server shall invoke the E-Get function to copy the Input String entered by the operator into the &inputBuffer.

If the Echo parameter of the Input service indication is true, the value contained in the &inputBuffer shall be copied into the &OutputBuffers for display by the D-Put function. This parameter shall be ignored for Operator Station objects of type **entry**.

The contents of the &inputBuffer shall be placed in the Input String parameter and a Result(+) shall be returned.

17.3 Output service

An MMS client may request the Output service to display data on an Operator Station object at the MMS server.

17.3.1 Structure

The structure of the component service primitives is shown in Table 99.

Table 99 - Output service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Operator Station Name	M	M(=)			
List of Output Data	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

17.3.1.1 Argument

This parameter shall convey the service specific parameters of the Output service request.

17.3.1.1.1 Operator Station Name

This parameter, of type Identifier, shall identify the Operator Station object to which output data is to be displayed.

17.3.1.1.2 List Of Output Data

This parameter shall be a list of character strings to be displayed. Each element of the list represents a single line on the display.

17.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

17.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

17.3.2 Service Procedure

17.3.2.1 Preconditions

The MMS server shall verify that:

a) the Operator Station identified by the Operator Station Name parameter exists.

b) the Operator Station has a &stationType field of **display** or **entry-display**.

c) all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = READ.

d) all the conditions in the Access Control List object referenced by the &access Control field of the Operator Station object are satisfied for Service Class = READ.

If any of these conditions is not met, the service shall fail and a Result(-) shall be returned.

17.3.2.2 Actions

The List Of Output Data parameter shall be copied into the &OutputBuffers field of the Operator Station object. The MMS server shall invoke the D-Put function to display these fields on the display.

A Result(+) shall be returned.

18 Event Management services

This clause defines no object models. It establishes the interworkings among the objects defined in the next 3 clauses.

This clause specifies the following services:

TriggerEvent	GetAlarmSummary
EventNotification	GetAlarmEnrollmentSummary
AcknowledgeEventNotification	AttachToEventCondition

The Event Management services provide facilities that allow an MMS client to define and manage events at a VMD and to obtain notifications of event occurrences. These facilities are provided in the VMD model by three (3) objects and by a set of nineteen (19) services and one (1) service modifier to operate upon these objects. These objects are the Event Condition object, the Event Action object, and the Event Enrollment object. Each of these objects models a specific aspect of the state information associated with the management of MMS Events. Only those events that have the potential of causing the initiation of an EventNotification service request are considered by this model. If the **cspi** CBB has been negotiated, an additional object, the Event Condition List, and six (6) additional services are included. This object and these services provide a convenience for dealing with large numbers of Event Condition objects.

This clause describes the Event Management model and the event detection and notification services. Clause 19 describes the Event Condition object and the services that act on it. Clause 20 describes the Event Action object and the services that act on it. The Event Enrollment object and the services that act on it are described in clause 21. The Event Condition List object and its services are described in clause 22.

The Event Condition object models that portion of the state information that is concerned with event detection and prioritization. It also includes information that assists in the determination of active "alarms".

The Event Action object models that portion of the state information that is concerned with the execution of MMS services upon the occurrence of an event.

The Event Enrollment object is used to relate the event occurrences of a given Event Condition object to notifications of these occurrences to a client and, optionally, to the execution of an Event Action. The Event Enrollment object includes state information that may be used for tracking and coordinating client responses to alarm event notifications. The Event Enrollment object may also be used for the delayed (conditional) execution of any confirmed service through the use of the Attach To Event Condition Modifier.

The interrelationship between these three objects is illustrated in Figure 17.

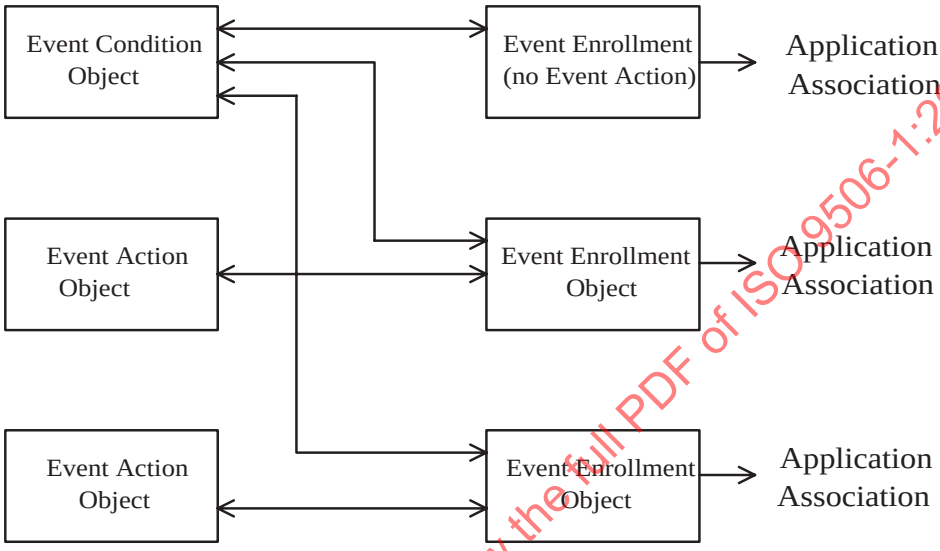

Figure 17 - Relationship Between Event Management Objects

Additional details on the Event Condition, Event Action and Event Enrollment objects, and the services and modifier that operate upon these objects, are provided in the clauses that follow.

18.1 Event Detection and Notification

An MMS server that supports the Event Management model shall be responsible for servicing events that occur at the VMD. This requires that the MMS server detect state changes for enabled monitored Event Condition objects, detect the deletion of referenced Event Condition objects, detect the occurrence of autonomous network-triggered events or accept requests to trigger network-triggered Event Condition objects, or any combination of the above actions, and process these changes (or TriggerEvent requests) in order to execute the Procedure for Event Transition Processing (see 18.1.1) for each client that has enrolled for the state change or for the TriggerEvent request.

Determination of the value of the &ecState field of an enabled monitored Event Condition object requires evaluation of the Event Condition object's state on a timely basis following a change in the determined value of the referenced boolean variable. In the case where the value of the Monitored Variable Reference attribute is equal to **unspecified**, the determination of the value of the &ecState field shall be a local matter. The decision as to when state evaluation is to be accomplished shall be a local (to the MMS server) matter. The &priority field and the &evaluationInterval field of the Event Condition object are provided as guidance in making this decision.

The procedure for Event Transition processing shall be executed:

a) when a change in the value of the &ecState field (to **active** or **idle**) of an enabled monitored Event Condition object is detected;

b) when the value of the &enabled field of a monitored Event Condition object is changed from true to false (by use of the AlterEventConditionMonitoring service, or by local means);

c) when a TriggerEvent service indication primitive is received for a network-triggered Event Condition object;

d) when an autonomous network-triggered event occurs;

e) when the &ecTransitions field of an Event Enrollment object contains the value **any-to-deleted** and the value of the &eventCondition field becomes undefined (as a result of deletion of the referenced Event Condition object); or,

f) when the &ecTransitions field of an Event Enrollment object contains the value **any-to-deleted** and the value of the &monitoredVariable field of a monitored Event Condition object, referenced by the &eventCondition field of the Event Enrollment object, becomes undefined (as a result of deletion of a Domain or loss of an application association).

NOTE The action to take when there are multiple transitions from different Event Condition objects to process concurrently, or when an additional transition of the Event Condition object currently being processed is detected during and prior to completion of the Procedure for Event Transition Processing, is a local matter. The &priority and &severity attributes are provided as guidance to the MMS server in this matter, and the &notificationLost field is provided as a means for the MMS server to declare to the client its inability to process completely the detected transitions.

It is possible that due to resource limitations at the MMS server, it may be temporarily impossible to complete a specific procedure or iteration. In this case, the MMS server may declare the procedure or iteration to have failed, or may suspend processing of the procedure or iteration until resources become available. The determination of whether to declare failure or temporarily suspend processing shall be a local matter. The decision criteria used in making this determination shall be reflected in the CSI as defined in clause 25 of ISO 9506-2.

18.1.1 Procedure for Event transition processing

This procedure shall involve the following actions:

NOTE 1 The mechanisms described in this clause describe the logical operation required for the Procedure for Event Transition Processing. Implementations may choose other methods of internal operation, as long as the externally visible characteristics described in this clause are maintained.

a) The Procedure for Event Condition Object Update (see 18.1.2) shall be executed.

Failure of the Procedure for Event Condition Object Update shall result in complete failure of the Procedure for Event Transition Processing. The value of the &notificationLost field for all Event Enrollment objects referenced by the &EventEnrollments field of the Event Condition object shall, if possible, be set to true.

b) The Procedure for Event Condition Object Attribute Value Capture (see 18.1.3) shall be executed.

Failure of the Procedure for Event Condition Object Attribute Value Capture shall result in complete failure of the Procedure for Event Transition Processing. The value of the &notificationLost field for all Event Enrollment objects referenced by the &EventEnrollments field of the Event Condition object shall, if possible, be set to true.

c) For each enrolled Event Enrollment object referenced by the Event Condition object's &EventEnrollments field that specifies the current transition of the Event Condition object in the value of its &ecTransitions field, the Procedure for Event Enrollment Object Attribute Value Capture (see 18.1.4) shall be executed.

Failure of any iteration of the Procedure for Event Enrollment Object Attribute Value Capture shall result in cancellation of the EventNotification service for the enrolled client. The value of the &notificationLost field of the specific Event Enrollment object shall, if possible, be set to true. If the &EventEnrollments field of the Event Condition object contains only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall fail.

d) For each Event Enrollment object referenced by the Event Condition object's &EventEnrollments field and specifying the current transition of the Event Condition object in the value of its &ecTransitions field, the Procedure for Event Enrollment Object Update (see 18.1.5) shall be executed.

Failure of any iteration of the Procedure for Event Enrollment object Update shall result in cancellation of the EventNotification service for the enrolled client. The value of the &notificationLost field of the specific Event Enrollment object shall, if possible, be set to true. If the &EventEnrollments field of the Event Condition object contains only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall fail.

e) For each Event Enrollment object referenced by the Event Condition object's &EventEnrollments field, specifying the current transition of the Event Condition object in the value of its &ecTransitions field and having a &eventAction field other than undefined, the Procedure for Event Action Execution (see 18.1.6) shall be executed.

If the value of the &eventAction field of the Event Enrollment object has become undefined as a result of Domain deletion or loss of an application association, the value of the Success Or Failure parameter of the EventNotification service request shall be false and the value of the Confirmed Service Error parameter of the EventNotification service request shall be OBJECT-UNDEFINED.

Failure of an iteration of the Procedure for Event Action Execution shall result in the value false appearing in the Success Or Failure parameter of the EventNotification service, when issued. Failure of the Procedure for Event Action Processing shall not otherwise affect the Procedure for Event Transition Processing.

f) For each Event Enrollment object referenced by the Event Condition object's &EventEnrollments field, specifying the value of the current transition in its &ecTransitions field, the Procedure for Establishment of an Application Association for Notification (see 18.1.7) shall be executed.

Failure of an iteration of the Procedure for Establishment of an Application Association for Notification shall result in cancellation of the EventNotification service for the enrolled client. The value of the &notificationLost field of the specific Event Enrollment object shall, if possible, be set to false. If the &EventEnrollments field of the Event Condition object should contain only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall fail.

g) For each Event Enrollment object referenced by the Event Condition object's &EventEnrollments field, specifying the value of the current transition in its &ecTransitions field, the Procedure for Invoking an Event Notification (see 18.1.8) shall be executed.

Failure of an iteration of the Procedure for Invoking an Event Notification shall result in cancellation of the EventNotification service for the enrolled client. The value of the &notificationLost field of the specific Event Enrollment object shall, if possible, be set to false. If the &EventEnrollments field of the Event Condition object contains only one reference to an Event Enrollment object, or if it is not possible to complete one iteration, the entire Procedure for Event Transition Processing shall fail.

NOTE 2 The actual invocation of the EventNotification service (including the execution of the applicable Event Action, if any) should occur as the captured information is processed, on a timely basis, in the sequence (within a given priority) in which it was captured.

Each of the above procedures shall be executed as an atomic entity, to the extent that a specific procedure shall succeed or fail as a discrete element or, where a procedure includes multiple iterations, each iteration shall succeed or fail as a discrete element. Failure of a specific procedure or iteration is defined as the inability to successfully complete the procedure, for any reason.

18.1.2 Procedure for Event Condition object update

This procedure shall be executed following determination of a change in the value of the &ecState field of the Event Condition object. The procedure involves the following actions:

a) Capture the time of determination (time of day or Time Sequence Identifier) of the new value of the &ecState field.

b) Alter the Event Condition object's &ecState field to the new state, as required.

c) If the Event Condition object is a monitored Event Condition object and the new state is **active**, replace the &timeToActive field with the captured time (date and time of day or Time Sequence Identifier).

d) If the Event Condition object is a monitored Event Condition object and the new state is **idle**, replace the &timeToIdle field with the captured time (date and time of day or Time Sequence Identifier).

18.1.3 Procedure for Event Condition object attribute value capture

This procedure shall be executed to capture the values of the attributes of the Event Condition object that are required in order to issue the EventNotification service. The values of the following attributes of the Event Condition object shall be captured:

a) the &name field of the Event Condition object;

b) the value of the &ecState field;

c) the value of the &severity field.

18.1.4 Procedure for Event Enrollment object attribute value capture

This procedure shall be executed for each Event Enrollment object specifying the value of the current transition in the &ecTransitions field. The values of the following attributes of the Event Enrollment object shall be captured:

a) the &name field of the Event Enrollment object;

b) the &aAssociation field;

c) the &notificationLost field;

d) the &eventAction field;

e) the &clientApplication field;

f) the &aaRule field.

The time of determination captured during the Procedure for Event Condition Object Update and the attribute values captured from the Event Condition and Event Enrollment objects shall be retained until the Procedure for Invoking the Event Notification Service (see 18.1.8) is completed.

18.1.5 Procedure for Event Enrollment object update

a) If the Event Enrollment object is a notification Event Enrollment object:

   1) if the Event Condition object's new &ecState field is **active**, replace the Event Enrollment object's &timeActiveAck field with the value **undefined**;

   2) if the Event Condition object's new &ecState field is **idle**, replace the Event Enrollment object's &timeIdleAck field with the value **undefined**;

b) if the Event Enrollment object is a modifier Event Enrollment object, continue processing of the Transaction object as defined in 7.4.3.

NOTE Transition processing should not wait for processing of the Transaction object.

18.1.6 Procedure for Event Action execution

The MMS server shall create a Transaction object (see 7.4.1) containing the following elements:

a) a unique locally assigned value for the &invokeID field;

b) a &Pre-ExecutionModifiers field equal to the Event Action object's &modifiers field;

c) a &currentModifier field initialized to refer to the first modifier in the &Pre-ExecutionModifiers;

d) a &confirmedService-Request field equal to the value of the Event Action object's &confirmedServiceRequest field;

e) a &Post-executionModifiers field that is empty;

f) a &cancelable field initialized to false.

A reference to this Transaction object shall be added to the &EATransitions field of the VMD. The procedure for transaction processing (see 7.4.3) shall be executed. The result of the execution of this procedure shall be used to supply the value of the Success Or Failure parameter in the Event Notification request service primitive and either the Confirmed Service Response parameter or the Confirmed Service Error parameter (depending on success or failure of the confirmed service) of the Action Result parameter of the EventNotification service request.

NOTE Transaction processing results in the consumption of resources and can potentially result in error conditions related to insufficient resources if care is not taken during application design. In addition, the Transaction object created, as a result of an Event Action execution, reduces the number of available outstanding confirmed service invocations, as limited by the maximum number of services outstanding parameters negotiated through the Initiate service as described in 8.2.1.2. If this limit is exceeded, the result of the Event Action (upon attempt to create the Transaction object) may be a service error specifying a resource problem.

18.1.7 Procedure for Establishment of an Application Association for notification

The application association to be used for transmission of EventNotification service primitives issued as a result of a notification Event Enrollment shall be established as specified below.

a) If the value of the &duration field of the Event Enrollment object is equal to **current**, the application association specified by the &aAssociation field of the Event Enrollment object shall be used for invoking the EventNotification service. If this application association terminates for any reason, the Event Enrollment object, and all pending event notifications for it, shall be deleted.

b) If the &duration field of the Event Enrollment object is equal to **permanent**, the application association to be used shall be determined as follows:

   1) The application association identified by the &aAssociation field of the Event Enrollment object shall be used if:

      i) the application association still exists;

      ii) the application association is with the client specified by the &clientApplication field of the Event Enrollment object;

      iii) the client supports reception of the EventNotification service.

   2) Otherwise, any currently active application association between the MMS server and the enrolled client shall be used, as long as reception of the EventNotification service by the enrolled client has been negotiated for the application association.

   3) If no suitable application association exists between the MMS server and the enrolled client, the MMS server shall attempt to establish a suitable application association with the enrolled client. If this succeeds, the newly established application association shall be used.

If it is not possible to establish an application association for the Event Enrollment, the MMS server may attempt to retry the establishment process on a periodic basis, or it may delete the Event Enrollment object and all pending event notifications requested by it. The action to be taken shall be a local decision of the MMS server.

NOTE Deletion of the Event Enrollment object should be performed as a last resort.

18.1.8 The Procedure for invoking an Event Notification

This procedure shall be executed in order to process the captured information and issue an EventNotification service request.

After determining the result of the execution of the Event Action (if applicable, see 18.1.6), and using the application association with the client (see 18.1.7), the EventNotification service request primitive shall be issued.

Following successful completion of the Procedure for Invoking an Event Notification, the value of the &notificationLost field of the specific Event Enrollment object shall be set to false. If necessary, the value of the &ackState field shall be updated.

NOTE There is an implied time skew between the time in the notification and the time of the values reported by the Event Action result. Implementations should attempt to minimize this interval.

18.2 TriggerEvent service

The TriggerEvent service is used by an MMS client to request that an MMS server trigger an event associated with a network-triggered Event Condition object.

18.2.1 Structure

The structure of the component service primitives is shown in Table 100.

Table 100 - TriggerEvent service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Condition Name	M	M(=)			
Priority	U	U(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

18.2.1.1 Argument

This parameter shall convey the parameters of the TriggerEvent service request.

18.2.1.1.1 Event Condition Name

This parameter, of type Object Name, shall specify the name of the network-triggered Event Condition object to be triggered.

18.2.1.1.2 Priority

If included, this parameter, of type Priority, shall contain a replacement value for the Event Condition object's &priority field. A change in priority shall take effect immediately, prior to execution of event transition processing.

If this parameter is omitted, the &priority field of the Event Condition object shall not be changed.

NOTE Depending on the particular use of priority by an implementation, it is possible that assignment of increased priority may result in a period during which EventNotification requests primitives are not sequentially ordered with respect to time of occurrence.

18.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

18.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

18.2.2 Service Procedure

18.2.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the Event Condition object are satisfied for the service class = LOAD.

The MMS server shall verify that the Event Condition object exists, and that the &ecClass field of this object has the value **network-triggered**.

If these conditions are not satisfied, the service shall fail and a Result(-) response shall be returned.

18.2.2.2 Actions

If the Priority parameter is present in the service request, the &priority field of the Event Condition object shall be changed to the value of the Priority parameter.

A Result(+) response shall be issued.

After the response is issued, the procedure for event transition processing (see 18.1.1) shall be executed for the specified Event Condition object.

18.3 EventNotification service

The EventNotification service is used by an MMS server to notify an enrolled client of the occurrence of a state transition associated with an Event Condition object. The EventNotification service is an unconfirmed service.

NOTE For this service, the MMS server issues the request primitive (see 26.2.1.1).

18.3.1 Structure

The structure of the component service primitives is shown in Table 101.

Table 101 - EventNotification service

Parameter Name	Req	Ind	CBB
Argument	M	M(=)	
Event Enrollment Name	M	M(=)	
Event Condition Name	M	M(=)	
Severity	M	M(=)	
Current State	C	C(=)	
Transition Time	M	M(=)	
Notification Lost	M	M(=)	
Alarm Acknowledgement Rule	C	C(=)	
Action Result	C	C(=)	
Event Action Name	M	M(=)	
Success or Failure	M	M(=)	
Confirmed Service Response	S	S(=)	
Confirmed Service Error	S	S(=)	
Display Enhancement	M	M(=)	cspi
String	S	S(=)	des
Index	S	S(=)	dei
No Enhancement	S	S(=)	

18.3.1.1 Argument

This parameter shall convey the parameters of the EventNotification service.

18.3.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Enrollment object for which this notification is invoked.

18.3.1.1.2 Event Condition Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object.

18.3.1.1.3 Severity

This parameter, of type integer, shall contain the value of the &severity field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object.

18.3.1.1.4 Current State

This parameter, of type EC-State, shall contain the value of the &ecState attribute, following event transition processing, of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object. If the EventNotification is being sent as a result of the transition **any-to-deleted** of the referenced Event Condition object, this parameter shall be omitted. This parameter shall also be omitted if the referenced Event Condition object has become unavailable, for example, as a result of deletion of a Domain or loss of an application association.

18.3.1.1.5 Transition Time

This parameter shall contain the time (date and time of day or Time Sequence Identifier) at which the Event Condition object transition was detected.

18.3.1.1.6 Notification Lost

This parameter, of type boolean, shall contain the value of the Event Enrollment object's &notificationLost field at the time of the Event Condition object's transition. If true, it indicates that one or more EventNotification requests specified for previous Event Condition object transitions associated with this Event Enrollment object have not been issued due to resource limitations in the issuing system. Following successful completion of an event notification, this parameter shall be set to the value false.

18.3.1.1.7 Alarm Acknowledgement Rule

This parameter, of type Alarm-Ack-Rule, shall only be included for EventNotification service requests resulting from an Event Condition object that has a value of the &ecClass field of **monitored**. It shall contain the value of the Event Enrollment object's &aaRule field.

This parameter shall be omitted for EventNotification service requests resulting from a network-triggered Event Condition object.

18.3.1.1.8 Action Result

This parameter shall be included for EventNotification service requests resulting from Event Enrollment objects that reference an Event Action object. Otherwise, it shall be omitted. If included, it shall contain the result of execution of the service request specified by the &confirmedServiceRequest field of the Event Action object. If the Event Action object has become unavailable, for example as a result of deletion of a Domain or loss of an application association, this parameter shall not be included. This parameter shall also not be included if the EventNotification is being sent as a result of the transition **any-to-deleted** of the referenced Event Condition object. The component parameters are specified as follows.

18.3.1.1.8.1 Event Action Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Action object referenced by the &eventAction field in the Event Enrollment object.

18.3.1.1.8.2 Success Or Failure

This boolean parameter shall indicate whether the execution of the confirmed service specified in the Event Action object's &confirmedServiceRequest field succeeded (true) or failed (false). Depending on the value of this parameter, one of the following parameters shall be selected.

18.3.1.1.8.3 Confirmed Service Response

This parameter shall contain the value of the Result(+) parameter resulting from the successful execution of the confirmed service specified in the &confirmedServiceRequest field of the Event Action object. It shall be present if the Success Or Failure parameter is true.

18.3.1.1.8.4 Confirmed Service Error

This parameter shall contain the value of the Result(-) parameter resulting from the failure in execution of the confirmed service. It shall be present if the Success Or Failure parameter is false.

18.3.1.1.9 Display Enhancement

This parameter shall be present only if the **cspi** CBB has been negotiated. If the &displayEnhancement field is present in the Event Enrollment object and its value is not **none**, this parameter shall indicate the value of the &displayEnhancement field of the Event Enrollment object. Otherwise, this parameter shall be the value of the &displayEnhancement field of the Event Condition object. Depending on the value of this parameter, one of the following parameters shall be selected.

18.3.1.1.9.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter.

18.3.1.1.9.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter.

18.3.1.1.9.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present.

18.3.2 Service Procedure

The MMS server shall issue an EventNotification request primitive as part of the Procedure for Event Transition processing (see 18.1.1). This International Standard does not specify a procedure for receiving the EventNotification service. The use of this service is application determined. However, an MMS-user receiving a EventNotification.indication primitive may choose to acknowledge the receipt of the notification (see 18.4). An EventNotification.request shall not be sent if the peer MMS-user did not indicate support of the EventNotification service in the Services Supported parameter of the Initiate service.

NOTE The MMS server, due to failure to receive a required acknowledgement or other local reason, may, at its discretion, repeat an EventNotification (in a new service invocation) for any Event Enrollment object that is awaiting a required acknowledgement. The criteria for deciding to repeat an EventNotification is a local matter.

18.4 AcknowledgeEventNotification service

The AcknowledgeEventNotification service is used by an MMS client to notify the MMS server that its user (usually a human operator) has acknowledged an EventNotification received from the MMS server.

18.4.1 Structure

The structure of the component service primitives is shown in Table 102.

Table 102 - AcknowledgeEventNotification service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Enrollment Name	M	M(=)			
Acknowledged State	M	M(=)			
Time Of Acknowledged Transition	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

18.4.1.1 Argument

This parameter shall convey the parameters of the AcknowledgeEventNotification service request.

18.4.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall be equal to the Event Enrollment Name parameter of the EventNotification that is being acknowledged.

18.4.1.1.2 Acknowledged State

This parameter, of type EC-State, shall be equal to the Current State parameter of the EventNotification that is being acknowledged.

18.4.1.1.3 Time Of Acknowledged Transition

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall be equal to the Transition Time parameter of the EventNotification that is being acknowledged.

18.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

18.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

18.4.2 Service Procedure

18.4.2.1 Preconditions

The MMS server shall verify that:

a) all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

b) all the conditions in the Access Control List object referenced by the &accessControl field of the Event Enrollment object are satisfied for the service class = LOAD.

If any of these conditions is not satisfied, the service shall fail and a Result(-) response shall be returned.

18.4.2.2 Actions

The Procedure For Acknowledgement of Event Notifications (see 18.4.2.3) shall be executed. The MMS server shall issue a Result(+) response.

18.4.2.3 Procedure For Acknowledgement of Event Notifications

a) If the attributes of the Event Enrollment object indicate that an acknowledgement for the specified state change (both state and time) has already been received, this procedure shall be skipped.

NOTE This may occur due to the possibility of multiple invocations of the EventNotification being issued by the MMS server to multiple clients.

b) Otherwise, if the Acknowledged State parameter is not equal to the current value of the &ecState field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object, or if the Time Of Acknowledged Transition parameter is not equal to the current value of the &timeToActive field or the &timeToIdle field (as applicable to the Acknowledged State) of the Event Condition object referenced by the acknowledged Event Enrollment object, this procedure shall be skipped.

c) Otherwise, if the value of the Acknowledged State parameter is ACTIVE, the acknowledged Event Enrollment object's &timeActiveAck field shall be replaced by the current time (date and time of day or Time Sequence Identifier).

d) Otherwise, if the value of the Acknowledged State parameter is IDLE, the acknowledged Event Enrollment object's &timeIdleAck field shall be replaced by the current time (date and time of day or Time Sequence Identifier).

e) If the value of the &aaRule field of the Event Enrollment object is not equal to **simple** or **none**, and the acknowledged transition corresponds to a required acknowledgement, the requirement shall be considered satisfied.

18.5 GetAlarmSummary service

The GetAlarmSummary service is used by an MMS client to request summary information from the MMS server about the current status of monitored Event Condition objects and related attributes of their referenced notification Event Enrollment objects.

18.5.1 Structure

The structure of the component service primitives is shown in Table 103.

Table 103 - GetAlarmSummary service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Enrollments Only	M	M(=)			
Active Alarms Only	M	M(=)			
Acknowledgement Filter	M	M(=)			
Severity Filter	M	M(=)			
Most Severe	M	M(=)			
Least Severe	M	M(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List Of Alarm Summary			M	M(=)	
Event Condition Name			M	M(=)	
Severity			M	M(=)	
Current State			M	M(=)	
Unacknowledged State			M	M(=)	
Display Enhancement			C	C(=)	cspi
String			S	S(=)	des
Index			S	S(=)	dei
No Enhancement			S	S(=)	
Time of Last Transition to Active			C	C(=)	
Time of Last Transition to Idle			C	C(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

18.5.1.1 Argument

This parameter shall convey the parameters of the GetAlarmSummary service request.

18.5.1.1.1 Enrollments Only

This parameter, of type boolean, shall provide a means of restricting the set of Event Condition objects that are to be summarized. When true, the summary shall include only those monitored Event Condition objects that contain (in the &EventEnrollments field) a reference to one or more notification Event Enrollment objects that specify (in the value of the &clientApplication field) the MMS client requesting the GetAlarmSummary service.

18.5.1.1.2 Active Alarms Only

This parameter, of type boolean, shall provide a means of restricting the Event Condition objects that are to be summarized. When true, only those **monitored** Event Condition objects that have the value of the &ecState field equal to **active** shall be included. When false, **monitored** Event Condition objects shall be summarized without regard to the value of the &ecState field.

18.5.1.1.3 Acknowledgement Filter

This parameter shall provide a means of restricting the Event Condition objects and Event Enrollment objects that are to be summarized. It may take any of three values:

NOTE In the following discussions, an unacknowledged event enrollment is a notification Event Enrollment object whose &ackState field is **noAckI** or **noAckA**, and an acknowledged event enrollment is a notification Event Enrollment object whose &ackState field is **acked**.

a) NOT-ACKED - the GetAlarmSummary response shall contain reports only for those **monitored** Event Condition objects for which the &EventEnrollments field contains a reference to at least one unacknowledged Event Enrollment.

b) ACKED - the GetAlarmSummary response contain reports only for those **monitored** Event Condition objects for which all references contained in the &EventEnrollments field are to acknowledged Event Enrollments.

c) ALL - the GetAlarmSummary response contain **monitored** Event Condition objects without regard to the acknowledgement status of any referenced notification Event Enrollment objects.

18.5.1.1.4 Severity Filter

This parameter, containing two integers, is provided as a means of restricting the summary to **monitored** Event Condition objects having specific severity. Only those Event Condition objects for which the value of the &severity field is between Most Severe and Least Severe, inclusive, shall be summarized.

18.5.1.1.5 Continue After

This parameter shall be included if requesting continuation of a partially completed alarm summary, as indicated by a value of true in the More Follows parameter in the most recently received GetAlarmSummary confirm primitive. Otherwise, this parameter shall be omitted.

This parameter, of type Object Name, shall be equal to the Event Condition Name parameter of the last Alarm Summary entry in the List Of Alarm Summary parameter of the GetAlarmSummary confirm service primitive for which continuation is requested.

18.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

18.5.1.2.1 List Of Alarm Summary

This parameter shall contain a list of zero or more Alarm Summary parameters, each providing the summary of a single **monitored** Event Condition object satisfying the criteria of the GetAlarmSummary service indication.

18.5.1.2.1.1 Event Condition Name

This parameter, of type Object Name, shall contain the value of the &name field of the monitored Event Condition object.

18.5.1.2.1.2 Severity

This parameter, of type integer, shall contain the value of the &severity field of the monitored Event Condition object.

18.5.1.2.1.3 Current State

This parameter, of type EC-State, shall contain the current value of the &ecState field of the monitored Event Condition object.

18.5.1.2.1.4 Display Enhancement

This parameter shall be present only if the **cspi** CBB has been negotiated. If the &displayEnhancement field is present in the Event Enrollment object and its value is not **undefined**, this parameter shall indicate the value of the &displayEnhancement field of the Event Enrollment object. Otherwise, this parameter shall be the value of the &displayEnhancement field of the Event Condition object. Depending on its value, one of the following parameters shall be selected.

18.5.1.2.1.4.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter.

18.5.1.2.1.4.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter.

18.5.1.2.1.4.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present.

18.5.1.2.1.5 Unacknowledged State

This parameter shall contain a value indicating the state of the Event Enrollment objects that reference this Event Condition object with respect to event notification acknowledgments. This parameter shall contain one of the following values:

a) NONE - indicates that no Event Enrollment object has an acknowledgement outstanding.

b) ACTIVE - indicates that at least one Event Enrollment object has an acknowledgement outstanding for the most recent transition to the **active** state.

c) IDLE - indicates that at least one Event Enrollment object has an acknowledgement outstanding for the most recent transition to the **idle** state.

d) BOTH - indicates that at least one Event Enrollment object has an acknowledgement outstanding for the most recent transition to the **active** state and at least one Event Enrollment object has an acknowledgement outstanding for the most recent transition to the **idle** state.

18.5.1.2.1.6 Time Of Last Transition To Active

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the &timeToActive field of the monitored Event Condition object, unless it has value **undefined**, in which case this parameter shall be omitted.

18.5.1.2.1.7 Time Of Last Transition To Idle

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the &timeToIdle field of the monitored Event Condition object, unless it has value **undefined**, in which case this parameter shall be omitted.

18.5.1.2.2 More Follows

The More Follows parameter, of type boolean, shall be true if this response does not contain all of the alarm summaries requested and a continuation request is needed to obtain the additional summaries. Otherwise, it shall be false.

18.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

18.5.2 Service Procedure

The MMS server shall prepare a list of monitored Event Condition objects for which the following conditions hold true:

a) the value of the &ecClass field is **monitored**;

b) the value of the &alarmSummaryReports field is true;

c) the filter criteria described in the Argument parameter are satisfied.

The list shall be in the collation sequence specified in 5.4.2.

The MMS server shall return a Result(+) response containing the List Of Alarm Summary parameter, constructed from zero or more monitored Event Condition objects from this list, and the More Follows parameter.

If the request contains the Continue After parameter, the List of Alarm Summary parameter shall begin with the first monitored Event Condition object after the monitored Event Condition object specified by the value of this parameter.

The number of monitored Event Condition objects that may be summarized in the List of Alarm Summary parameter may be limited by local restrictions. If the response does not contain all of the requested alarm summaries, the More Follows parameter shall be set to true. Otherwise, the More Follows parameter shall be set to false.

If the More Follows parameter is true, there shall be at least one Event Condition object reported in the List of Alarm Summary parameter.

18.6 GetAlarmEnrollmentSummary service

The GetAlarmEnrollmentSummary service is used by an MMS client to request summary information from the MMS server about the current alarm status of notification Event Enrollment objects and related attributes of their referenced monitored Event Condition objects.

18.6.1 Structure

The structure of the component service primitives is shown in Table 104.

Table 104 - GetAlarmEnrollmentSummary service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Enrollments Only	M	M(=)			
Active Alarms Only	M	M(=)			
Acknowledgement Filter	M	M(=)			
Severity Filter	M	M(=)			
Most Severe	M	M(=)			
Least Severe	M	M(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List Of Enrollment Summary			M	M(=)	
Event Enrollment Name			M	M(=)	
Client Application			C	C(=)	
Severity			M	M(=)	
Current State			M	M(=)	
Display Enhancement			M	M(=)	cspi
String			S	S(=)	des
Index			S	S(=)	dei
No Enhancement			S	S(=)	
Notification Lost			M	M(=)	
Alarm Acknowledgement Rule			M	M(=)	
Enrollment State			C	C(=)	
Last Transition to Active			C	C(=)	
Time Active Acknowledged			C	C(=)	
Last Transition to Idle			C	C(=)	
Time Idle Acknowledged			C	C(=)	
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

18.6.1.1 Argument

This parameter shall convey the parameters of the GetAlarmEnrollmentSummary service request.

18.6.1.1.1 Enrollments Only

This parameter, of type boolean, shall provide a means of restricting the set of Event Enrollment objects to be summarized. When true, only those **notification** Event Enrollment objects for which the value of the &clientApplication field specifies the requesting MMS client shall be summarized. When false, notification Event Enrollment objects shall be summarized without regard to the value of the &clientApplication field.

18.6.1.1.2 Active Alarms Only

This parameter, of type boolean, shall provide a means of restricting the **notification** Event Enrollment objects to be summarized. When true, only those **notification** Event Enrollment objects that reference an Event Condition object whose &ecState field value is **active** shall be included. When false, **notification** Event Enrollment objects shall be summarized without regard to state of the Event Condition object.

18.6.1.1.3 Acknowledgement Filter

This parameter, of type integer, shall provide a means of restricting the Event Enrollment objects to be summarized. It may take any of three values:

NOTE In the following discussions, an unacknowledged Event Enrollment is a notification Event Enrollment object whose &ackState field has the value **noAckI** or the **noAckA**, and an acknowledged Event Enrollment is a notification Event Enrollment object whose &ackState field has the value **acked**.

a) NOT-ACKED - requests that the GetAlarmEnrollmentSummary response contain only reports for unacknowledged Event Enrollments;

b) ACKED - requests that the GetAlarmEnrollmentSummary response contain only reports for acknowledged Event Enrollments;

c) ALL - requests that the GetAlarmEnrollmentSummary response contain reports of Event Enrollments for all **notification** Event Enrollment objects without regard to acknowledgement status.

18.6.1.1.4 Severity Filter

This parameter, containing two integers, is provided as a means of restricting the summary to **notification** Event Enrollment objects referencing Event Condition objects having specific severity. Only **notification** Event Enrollment objects referencing Event Condition objects for which the value of the &severity field is between Most Severe and Least Severe, inclusive, shall be summarized.

18.6.1.1.5 Continue After

This parameter, of type Object Name, shall be included if requesting continuation of a partially completed alarm enrollment summary, as indicated by a value of true in the More Follows parameter in the most recently received GetAlarmEnrollmentSummary confirm primitive. Otherwise, this parameter shall be omitted. When included, this parameter shall contain the value of the last Event Enrollment Name parameter included in the List Of Alarm Enrollment Summary response primitive preceding this request.

18.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

18.6.1.2.1 List Of Enrollment Summary

This parameter shall contain a list of zero or more Alarm Enrollment Summary parameters, each providing the summary of a single **notification** Event Enrollment object satisfying the criteria of the GetAlarmEnrollmentSummary service indication.

18.6.1.2.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the &name field of the **notification** Event Enrollment object.

18.6.1.2.1.2 Client Application

This parameter, of type Application Reference, shall contain the value of the &clientApplication field of the **notification** Event Enrollment object. If this field specifies the requesting MMS client, this parameter shall be omitted.

18.6.1.2.1.3 Severity

This parameter, of type integer, shall contain the value of the &severity field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object.

18.6.1.2.1.4 Current State

This parameter, of type EC State, shall contain the value of the &ecState field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object.

18.6.1.2.1.5 Display Enhancement

This parameter shall be present only if the **cspi** CBB has been negotiated. If the &displayEnhancement field is present in the Event Enrollment object and its value is not **undefined**, this parameter shall indicate the value of the &displayEnhancement field of the Event Enrollment object. Otherwise, this parameter shall be the value of the &displayEnhancement field of the Event Condition object. Depending on its value, one of the following parameters shall be selected.

18.6.1.2.1.5.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter.

18.6.1.2.1.5.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter.

18.6.1.2.1.5.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present.

18.6.1.2.1.6 Notification Lost

This boolean parameter shall contain the value of the &notificationLost field of the Event Enrollment object.

18.6.1.2.1.7 Alarm Acknowledgement Rule

This parameter, of type AlarmAckRule, shall contain the value of the &aaRule field of the Event Enrollment object.

18.6.1.2.1.8 Enrollment State

This parameter, of type EE-State, depends on the value of the &aaRule field of the Event Enrollment object and shall reflect the value of the &ecState field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object and the value of the &ackState field of the Event Enrollment object. The value of this parameter is prescribed in 21.5.1.2.5.

18.6.1.2.1.9 Last Transition To Active

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the &timeToActive field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment. If the &timeToActive field has the value **undefined**, this parameter shall be omitted.

18.6.1.2.1.10 Time Active Acknowledged

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the &timeActiveAck field of the Event Enrollment object. If the &timeActiveAck field has the value **undefined**, this parameter shall be omitted.

18.6.1.2.1.11 Last Transition To Idle

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the &timeToIdle field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment. If the &timeToIdle field has the value **undefined**, this parameter shall be omitted.

18.6.1.2.1.12 Time Idle Acknowledged

This parameter, expressed as a date and time of day or Time Sequence Identifier, shall contain the value of the &timeIdleAck field of the Event Enrollment object. If the &timeIdleAck field has the value **undefined**, this parameter shall be omitted.

18.6.1.2.2 More Follows

The More Follows parameter, of type boolean, shall have a value true if this response does not contain all of the alarm enrollment summaries requested and a continuation request is required to obtain the additional summaries. Otherwise, the value shall be false.

18.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

18.6.2 Service Procedure

The MMS server shall prepare a list of Event Enrollment objects for inclusion in the List Of Alarm Enrollment Summary parameter. This list shall be prepared by (1) assembling a list of all **monitored** Event Condition objects, ordered by the collation sequence specified in 5.4.2. For each such Event Condition the **notification** Event Enrollment objects that satisfy the specified filter criteria and whose &aaRule field is not equal to **none** shall constitute the members of the list. For each Event Condition object, the Event Enrollment objects shall be ordered by the collation sequence specified in 5.4.2. The MMS server shall return a Result(+) response containing the List Of Alarm Enrollment Summary parameter constructed from this list and the More Follows parameter.

If the request contains the Continue After parameter, the List of Alarm Enrollment Summary parameter shall begin with the first notification Event Enrollment object after the notification Event Enrollment object specified by the value of this parameter.

The number of notification Event Enrollment objects that may be summarized in the List of Alarm Enrollment Summary parameter may be limited by local restrictions. If the response does not contain all of the requested alarm summaries, the More Follows parameter shall be set to true. Otherwise, the More Follows parameter shall be set to false.

If the More Follows parameter is true, there shall be at least one Event Enrollment object reported in the List of Alarm Enrollment Summary parameter.

18.7 Attach To Event Condition Modifier

The Attach To Event Condition modifier is used by an MMS client to request that the MMS server delay execution of a requested service's service procedure until a specified Event Condition object undergoes any one of a specified set of state transitions.

18.7.1 Structure

The structure of the Attach To Event Condition modifier parameter is shown in Table 105.

Table 105 - Attach To Event Condition Modifier

Parameter Name	Req	Ind
Attach To Event Condition	S	S(=)
Event Enrollment Name	M	M(=)
Event Condition Name	M	M(=)
Causing Transitions	M	M(=)
Acceptable Delay	U	U(=)

18.7.1.1 Attach To Event Condition

The Attach To Event Condition parameter is provided as an parameter of the List of Modifier parameter for each confirmed service request (see 5.6). The sub-parameters of the Attach To Event Condition modifier are specified as follows.

18.7.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the &name field used to identify the modifier Event Enrollment object. This name shall be unique among all other &name fields of Event Enrollment objects of identical scope at the VMD.

18.7.1.1.2 Event Condition Name

This parameter, of type Object Name, shall specify the &name field of the Event Condition object that shall be the &eventCondition field of the modifier Event Enrollment object.

18.7.1.1.3 Causing Transitions

This parameter, of type Transitions, shall specify the set of transitions of the Event Condition object that are to cause processing of the service request to continue.

18.7.1.1.4 Acceptable Delay

This optional parameter, of type integer, shall indicate the value of the &remainingDelay field of the Event Enrollment object. If this parameter is not present, the value of &remainingDelay shall be **forever**.

If more than one Attach To Event Condition modifier has been specified for this service request, this parameter applies only to the modifier for which it is a parameter. Each such modifier may have different value for this parameter.

18.7.2 Service Procedure

18.7.2.1 Preconditions

If the Acceptable Delay parameter is present and its value is zero, a Result(-) response shall be issued.

18.7.2.2 Action Step 1

The MMS server shall create a **modifier** Event Enrollment object initialized as specified below and shall place a reference to this Event Enrollment object in the &EventEnrollments field of the specified Event Condition object. No further action shall be taken toward execution of the indicated service's service procedure until the specified Event Condition object undergoes one of the specified state transitions.

The created Event Enrollment object shall be initialized as follows:

a) The &name field of the Event Enrollment object shall be set to the value provided in the Event Enrollment Name parameter.

b) The &eeClass field shall be initialized to the value **modifier**.

c) The &eventCondition field shall be initialized to reference the Event Condition object identified by the value of the Event Condition Name parameter.

d) The &ecTransitions field shall be initialized to the value of the Causing Transitions parameter.

e) The &invokeID field shall be initialized to contain the &invokeID field of the Transaction object that carries this service request (see 7.4.1).

f) If no value has been provided for the Acceptable Delay parameter, the &remainingDelay field shall be set to the value **forever**. If a value has been provided, the &remainingDelay field shall be initialized to the value of the Acceptable Delay parameter and the timer function activated.

g) The &aAssociation field shall be initialized to identify the application association that was used to execute this service.

18.7.2.3 Action Step 2

If the timer function had been activated and the value of the &remainingDelay field reaches zero, the following actions shall be performed:

a) The timer shall be deactivated.

b) The MMS server shall return a Result(-) response for the modified service with the error parameter indicating expiration of acceptable delay;

c) The procedure for Event Enrollment Deletion (see 21.3.3) shall be executed.

18.7.2.4 Action Step 3

If a specified state transition of the Event Condition object is detected, the following actions shall be performed:

a) The service request shall be released for continued execution of its service procedure (see 7.4.3);

b) The Event Enrollment object shall be deleted according to the Procedure for Event Enrollment Deletion (see 21.3.3).

18.7.3 Procedure for Cancellation of Modified Service

This procedure amends the service procedure for the Cancel service if a service request attached to an Event Condition object is to be cancelled. The procedure is as follows:

a) remove the reference to the Event Enrollment object from the &EventEnrollments field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object;

b) execute the service procedure specified for the Cancel service; then

c) delete the Event Enrollment object using the Procedure For Event Enrollment Deletion (see 21.3.3).

NOTE Cancellation of the modified service results in a negative response being issued for the service invocation.

18.8 Conformance Requirements Unique to Event Management

The Event Management Services define parameter and time-support requirements for the MMS server. These requirements are described below.

18.8.1 Parameter Conformance Building Blocks

See 8.1.3.16 for a description of the **cei** parameter CBB.

18.8.2 Support for Time

The Configuration and Initialization Statement (CSI) for an implementation shall state the level of support for time (date and time of day, or Time Sequence Identifier).

Support for the Time Sequence Identifier implies that an implementation will be capable of assigning a sequence number to all attributes that represent time, but that real time in terms of date and time of day is not supported. The relationship of the Time Sequence Identifier to date and time of day shall be for local determination.

Support for date and time of day implies that an implementation maintains a real time clock that is used to assign the values for all attributes that contain time (date and time of day).

19 Event Condition services

19.1 Event Conditions

This clause provides an object model for the following object:

EVENT-CONDITION

This clause specifies the following services:

DefineEventCondition
DeleteEventCondition
GetEventConditionAttributes
ReportEventConditionStatus
AlterEventConditionMonitoring

The Event Condition object models the MMS-visible aspects of an Event Condition. The event management model defines two classes of Event Condition objects. They are the **network-triggered** Event Condition object and the **monitored** Event Condition object.

A **network-triggered** Event Condition object models an event occurring due to the explicit request of an MMS client using the TriggerEvent service. The **network-triggered** Event Condition object also models events that occur internally, as a result of autonomous MMS server actions.

A **monitored** Event Condition object is a virtual representation of an aspect of MMS server activity that, due to application design criteria, represents a significant occurrence in the processing of the MMS server. Events associated with a **monitored** Event Condition object are detected by the autonomous action of the MMS server.

Either class of Event Condition object includes the potential for MMS server initiation of the EventNotification service, through the Procedure for Event Transition Processing (see 18.1.1). The model of an Event Condition object is specified below, followed by a description of the services that operate on the Event Condition object.

19.1.1 The Event Condition object

```
EVENT-CONDITION ::= CLASS {
    &name                       ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
    &accessControl              ACCESS-CONTROL-LIST,
    &ecClass                    EC-Class,
    &ecState                    EC-State,
    &priority                   Priority,
    &severity                   Severity,
    &EventEnrollments           EVENT-ENROLLMENT,
-- The following fields shall be present
-- if and only if the value of &ecClass is monitored.
    &enabled                    BOOLEAN OPTIONAL,
    &alarmSummaryReports        BOOLEAN OPTIONAL,
    &monitoredVariable          CHOICE {
        named                       NAMED-VARIABLE,
        unnamed                     UNNAMED-VARIABLE,
        unspecified                 NULL-OBJECT } OPTIONAL,
    &evaluationInterval         INTEGER OPTIONAL,
    &timeToActive               EventTime OPTIONAL,
    &timeToIdle                 EventTime OPTIONAL,
IF (cspi)
    &displayEnhancement         CHOICE {
IF (des)
```

```
        text                                MMSString,
ENDIF
IF (dei)
        number                              INTEGER,
ENDIF
        none                                NULL
        } OPTIONAL,
    &group-Priority-Override            CHOICE {
        priority                            Priority,
        undefined                           NULL
        } OPTIONAL,
    &ReferencingEventConditionLists     EVENT-CONDITION-LIST OPTIONAL
ENDIF
    }
```

19.1.1.1 &name

The &name field uniquely identifies the Event Condition object within the VMD. The &name may have VMD-specific, Domain-specific or AA-specific scope.

19.1.1.2 &accessControl

The &accessControl field identifies an Access Control List object that provides conditions under which this Event Condition may have its monitoring attributes modified, may be triggered (if a network-triggered Event Condition), have its access control changed, or be deleted.

19.1.1.3 &ecClass

The &ecClass field indicates the class of the Event Condition object. This field contains the value **network-triggered**, or the value **monitored**.

```
EC-Class ::= INTEGER {
    network-triggered       (0),
    monitored               (1) }
```

19.1.1.4 &ecState

The &ecState field shall specify the state of the Event Condition object. This field contains the value **disabled**, **idle**, or **active**. The value of this field is always **disabled** for a **network-triggered** Event Condition object.

```
EC-State ::= INTEGER {
    disabled                (0),
    idle                    (1),
    active                  (2) }
```

19.1.1.5 &priority

The &priority field indicates the priority of the Event Condition object relative to other Event Condition objects defined at the VMD. The &priority field is an integer with values ranging from zero (0) to one hundred twenty seven (127), inclusive. Zero represents the highest priority; one hundred twenty seven (127) represents the lowest priority. Sixty four (64) represents normal priority. Behaviour of the MMS server with respect to the value of the &priority field is a local issue.

NOTE Priority and normalPriority are defined in 16.1.3.7.

19.1.1.6 &severity

The &severity field represents the effect of the event on the process being controlled. The &severity field is an integer with values ranging from zero (0) to one hundred twenty seven (127), inclusive. Zero indicates the most severe; one hundred twenty seven

indicates the least severe. Sixty four (64) represents normal severity. Behaviour of the MMS server with respect to the value of the &severity field is a local issue.

Severity ::= Integer8

normalSeverity Severity ::= 64

19.1.1.7 &EventEnrollments

The &EventEnrollments field identifies a set of zero or more Event Enrollment objects, each of which refers to this Event Condition object.

19.1.1.8 &enabled

The &enabled field exists only for **monitored** Event Conditions. It specifies whether (true) or not (false) changes in the value of the variable referenced by the Event Condition object's &monitoredVariable field shall cause invocation of the Procedure for Event Transition Processing for the Event Enrollment objects specified in the &EventEnrollments field.

19.1.1.9 &alarmSummaryReports

The &alarmSummaryReports field exists only for **monitored** Event Condition objects. If true, it indicates that the Event Condition object shall be considered for inclusion in responses to the GetAlarmSummary service without regard to the value of the &ecState field of the Event Condition object and the value of the &aaRule field of any referenced Event Enrollment objects. If false, the Event Condition object shall not be considered for inclusion in alarm summaries unless at least one referenced Event Enrollment object contains a value for the &aaRule field that is not equal to **none**.

19.1.1.10 &monitoredVariable

The &monitoredVariable field exists only for **monitored** Event Conditions. It identifies a variable object (Unnamed or Named) of type boolean or the Null object. The value determined through the use of the referenced variable object is monitored (by the MMS server through the V-Get function) to determine the value of the &ecState field of an enabled **monitored** Event Condition object. For a locally defined Event Condition, or for an Event Condition created through the use of the CreateProgramInvocation service, this field shall specify the Null object, indicating that the event transitions are determined by conditions other than the value of an MMS-visible variable object.

NOTE The relationship between the value determined through the use of the referenced monitored variable object and the state of the process controlled by the MMS server is a local issue.

19.1.1.11 &evaluationInterval

The &evaluationInterval field exists only for a **monitored** Event Condition. The &evaluationInterval field shall specify the maximum acceptable time, in milliseconds, between successive determinations of the value of the Event Condition object's &ecState field.

19.1.1.12 &timeToActive

The &timeToActive field exists only for a **monitored** Event Condition object. It shall specify the time (date and time of day or Time Sequence Identifier) of the last detected transition of the value of the Event Condition object's &ecState to **active**. If the Event Condition object's &ecState field has never had the value **active**, this field shall have the value **undefined**.

```
EventTime ::= CHOICE {
    timeOfDay                [0] IMPLICIT TimeOfDay,
    timeSequenceIdentifier   [1] IMPLICIT Unsigned32,
    undefined                [2] IMPLICIT NULL }
```

19.1.1.13 &timeToIdle

The &timeToIdle field exists only for a monitored Event Condition object. It shall specify the time (date and time of day or Time Sequence Identifier) of the last detected transition of the Event Condition object's &ecState field to the value **idle**. If the Event Condition object's &ecState field has never had the value **idle**, this field shall have the value **undefined**.

19.1.1.14 &displayEnhancement

This field specifies the type of the &displayEnhancement field of the Event Condition object. This field is present only if the **cspi** parameter CBB has been negotiated. If the value of this field is **text**, the &displayEnhancement field is of type character string. If the value of this field is **number**, the &displayEnhancement is of type integer. If the value of this field is **none**, the &displayEnhancement field is null.

19.1.1.15 &group-Priority-Override

This field is present only if the **cspi** parameter CBB has been negotiated. This field shall contain a value that is either null or an integer value between zero (0) and one hundred twenty-seven (127). Zero shall represent the highest priority and one hundred twenty-seven shall represent the lowest priority. If the value of this field is defined, it shall represent a priority value to be used by the MMS server in place of the value contained in the &priority field. If the value of the &group-Priority-Override field is **undefined**, the MMS server shall use the value of the &priority field in determining the importance of the Event Condition object.

This field need not be implemented if Event Condition List objects are not supported.

19.1.1.16 &ReferencingEventConditionLists

This field shall specify a set of Event Condition List objects that reference this Event Condition object. This field is present only if the **cspi** parameter CBB has been negotiated.

This field need not be implemented if Event Condition List objects are not supported.

19.1.2 Event Condition State Diagrams

The state diagram for an event condition depends on the class of the Event Condition.

19.1.2.1 Network-triggered Event Condition

The state diagram that describes the network-triggered Event Condition object is shown in Figure 18.

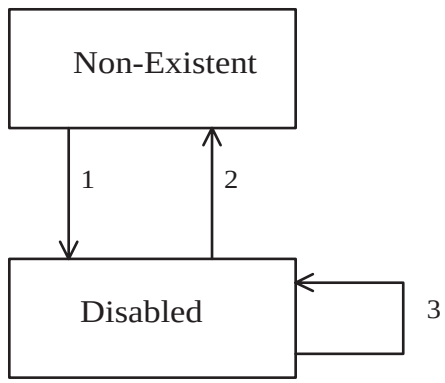

Figure 18 - Network-triggered Event Condition State Diagram

Transitions:

1) Receive DefineEventCondition indication specifying a **network-triggered** Event Condition object.

2) Receive DeleteEventCondition indication with the conditions for deletion of an Event Condition object being satisfied (see 19.3.2), or loss of application association if the Event Condition object is of AA-specific scope and dependent on the lost application association, or deletion of a Domain if the Event Condition object is of Domain-specific scope and dependent on the deleted Domain.

3) Receipt of any of the following indications:

- DeleteEventCondition with the conditions for deletion of an Event Condition object not being satisfied (see 19.3.2)
- GetEventConditionAttributes;
- ReportEventConditionStatus;
- AlterEventConditionMonitoring;
- TriggerEvent;
- GetEventEnrollmentAttributes;
- DefineEventEnrollment;
- DeleteEventEnrollment;
- AcknowledgeEventNotification;
- GetAlarmSummary;
- GetAlarmEnrollmentSummary; or
- Service modified by Attach To Event Condition;

19.1.2.2 Monitored Event Condition

The state diagram that describes a monitored event condition is defined by Figure 19.

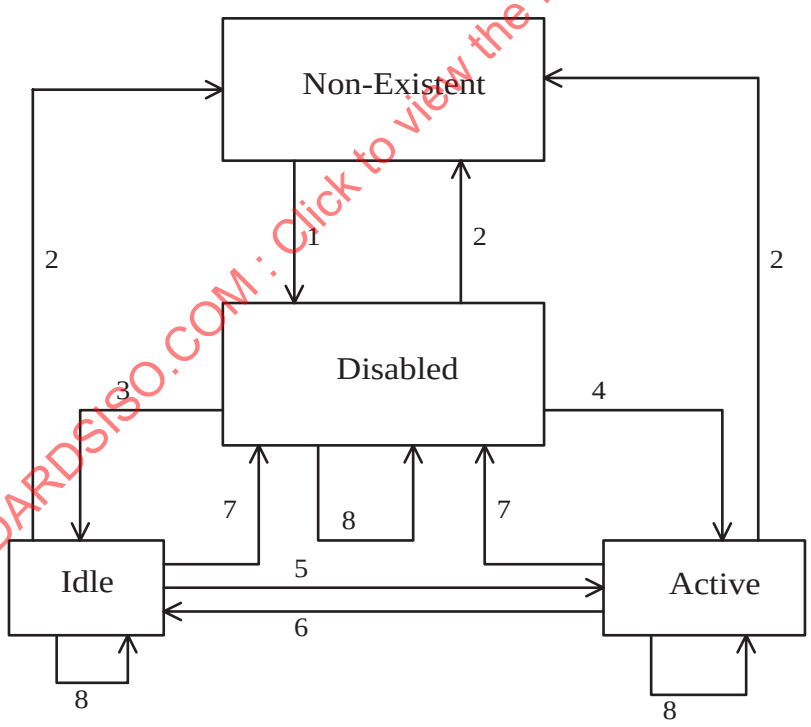

Figure 19 - Monitored Event Condition State Diagram

Transitions:

1) Receive DefineEventCondition indication specifying a **monitored** event condition.

2) Receive DeleteEventCondition indication with the conditions for deletion of an Event Condition object being satisfied (see 19.3.2)

3) Receive AlterEventConditionMonitoring indication with the &enable field equal to true while the value of the variable referenced by the &monitoredVariable field is false.

4) Receive AlterEventConditionMonitoring indication with the &enable field equal to true while the value of the variable referenced by the &monitoredVariable field is true.

5) Value of the variable referenced by the &monitoredVariable field changes from false to true.

6) Value of the variable referenced by the &monitoredVariable field changes from true to false.

7) Receive AlterEventConditionMonitoring indication with the &enable field equal to false.

8) Receipt of any of the following indications:

- DeleteEventCondition with the conditions for deletion of an Event Condition object not being satisfied (see 19.3.2);
- GetEventConditionAttributes;
- ReportEventConditionStatus;
- AlterEventConditionMonitoring with Enable parameter omitted or equal to current value of the &enable field;
- TriggerEvent;
- GetEventEnrollmentAttributes;
- DefineEventEnrollment;
- DeleteEventEnrollment;
- GetAlarmSummary;
- GetAlarmEnrollmentSummary; or
- Service modified by Attach To Event Condition;

19.2 DefineEventCondition service

The DefineEventCondition service is used by an MMS client to request the creation of an Event Condition object at a VMD.

19.2.1 Structure

The structure of the component service primitives is shown in Table 106.

Table 106 - DefineEventCondition service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Condition Name	M	M(=)			
Class	M	M(=)			
Priority	M	M(=)			
Severity	M	M(=)			
Alarm Summary Reports	C	C(=)			
Monitored Variable	C	C(=)			
Evaluation Interval	C	C(=)			
Display Enhancement	M	M(=)			cspi
String	S	S(=)			des
Index	S	S(=)			dei
No Enhancement	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

19.2.1.1 Argument

This parameter shall convey the parameters of the DefineEventCondition service request.

19.2.1.1.1 Event Condition Name

The Event Condition Name parameter, of type Object Name, shall contain the name to be assigned to the created Event Condition object. It shall uniquely identify the Event Condition object.

19.2.1.1.2 Class

The Class parameter, of type EC-Class, shall be the initial value of the &ecClass field of the newly created Event Condition object.

19.2.1.1.3 Priority

The Priority parameter, of type Priority, shall contain the initial value of the &priority field of the newly created Event Condition object.

19.2.1.1.4 Severity

The Severity parameter, of type Severity, shall contain the initial value of the &severity field of the newly created Event Condition object.

19.2.1.1.5 Alarm Summary Reports

This parameter, of type boolean, shall be present if and only if the Event Condition object to be created is a monitored Event Condition object. If specified, the Alarm Summary Reports parameter shall contain the initial value of the Event Condition object's &alarmSummaryReports field.

19.2.1.1.6 Monitored Variable

The Monitored Variable parameter, of type Variable Specification, shall be present if and only if the Event Condition object to be created is a **monitored** Event Condition object. If present, it shall specify a Named or Unnamed Variable object of boolean type that is to be referenced by the Event Condition object's &monitoredVariableReference field.

19.2.1.1.7 Evaluation Interval

The Evaluation Interval parameter, of type integer, shall be present if and only if the Event Condition object to be created is a **monitored** Event Condition object. It shall specify the maximum acceptable time, in milliseconds, between determinations of the value of the Event Condition object's &ecState field. The value of this parameter shall be such that determination at a periodic interval less than or equal to the specified interval shall be sufficient for reliable and timely detection of state changes. Any value between zero and 2**31 - 1, inclusive, may be specified.

This parameter shall be provided as guidance to the MMS server. Acceptance of this value guarantees only that the MMS server shall make every effort to honour the specified value. A MMS server that determines that it is unable to honour the service request due to resource or other limitations shall return a TIME-RESOLUTION error in response to the service request.

The actual decision to determine the value of the Event Condition object's &ecState field may be based on the passage of time (such as a scan cycle), on knowledge about the variable referenced by the &monitoredVariable field, or other local criteria. If the decision is to be time-based, a non-zero value for the Evaluation Interval parameter shall specify the maximum acceptable time between determinations. A zero value shall indicate that any interval is acceptable.

19.2.1.1.7.1 Display Enhancement

This parameter shall be present if and only if the **cspi** CBB has been negotiated. If this parameter is present, one of the following parameters shall be selected.

19.2.1.1.7.1.1 String

This choice indicates that the string form of the Display Enhancement parameter has been selected. This selection may be made only if the **des** CBB has been negotiated.

19.2.1.1.7.1.2 Index

This choice indicates that the numeric form of the Display Enhancement parameter has been selected. This selection may be made only if the **dei** CBB has been negotiated.

19.2.1.1.7.1.3 No Enhancement

This choice indicates that no Display Enhancement has been selected. This parameter shall be selected if neither **des** nor **dei** has been negotiated.

NOTE This choice may be made (user option) if **des** or **dei** have been negotiated.

19.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

19.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

19.2.2 Service Procedure

19.2.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD. If this conditions is not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED.

The MMS server shall verify that no Event Condition with a &name parameter equal to the Event Condition Name parameter already exists. If this condition is not satisfied, a Result(-) shall be returned.

19.2.2.2 Actions

A new Event Condition object shall be created and initialized as described below.

a) &name - Initialized to equal the value of the Event Condition Name parameter.

b) &accessControl - Initialized to reference an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) &ecClass - Initialized to equal the value of the Class parameter.

d) &ecState - Initialized to the value **disabled**.

e) &priority - Initialized to the value of the Priority parameter.

f) &severity - Initialized to the value of the Severity parameter.

g) &alarmSummaryReports - Initialized to the value of Alarm Summary Reports parameter, if present.

h) &EventEnrollments - Initialized to empty.

i) &enabled - Initialized to the value false.

j) &monitoredVariable - Initialized to reference the object specified by the value of the Monitored Variable parameter.

k) &evaluationInterval - Initialized to equal the value of the Evaluation Interval parameter.

l) &timeToActive - Initialized to the value **undefined**.

m) &timeToIdle - Initialized to the value **undefined**.

n) If the **cspi** CBB has been negotiated and if the String choice of Display Enhancement parameter has been selected, the value of the &displayEnhancement field of the Event Condition shall be the value of String parameter.

o) If the **cspi** CBB has been negotiated and if the Index choice of the Display Enhancement parameter has been selected, the value of &displayEnhancement field of the Event Condition shall be the value of the Index parameter.

p) If the **cspi** CBB has been negotiated and if the No Enhancement choice of the Display Enhancement parameter has been selected, the value of &displayEnhancement field of the Event Condition shall be **none**.

q) If the **cspi** CBB has been negotiated, the &group-Priority-Override field shall be initialized to **undefined**.

r) If the **cspi** CBB has been negotiated, the &ReferencingEventConditionLists field shall be initialized to empty.

A Result(+) shall be returned, indicating that the Event Condition object has been created.

19.3 DeleteEventCondition service

The DeleteEventCondition service is used by an MMS client to request that the MMS server delete one or more Event Condition objects.

19.3.1 Structure

The structure of the component service primitives is shown in Table 107.

Table 107 - DeleteEventCondition service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
Event Condition Names	S	S(=)			
AA Specific	S	S(=)			
Domain Name	S	S(=)			
VMD Specific	S	S(=)			
Result(+)			S	S(=)	
Candidates Not Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

19.3.1.1 Argument

This parameter shall convey the parameters of the DeleteEventCondition service request.

19.3.1.1.1 Scope Of Delete

The Scope Of Delete parameter shall specify the extent of delete to be attempted. One of the following parameters shall be selected.

19.3.1.1.1.1 Event Condition Names

Selection of this parameter, containing a list of one or more Event Condition Name parameters, each of type Object Name and each identifying an Event Condition object, indicates that a specific list of Event Condition objects is to be deleted. This parameter provides the names of the specific candidate Event Condition objects to be deleted.

19.3.1.1.1.2 AA Specific

Selection of this parameter shall indicate that all Event Condition objects whose scope is the current application association are candidates for deletion.

19.3.1.1.1.3 Domain Name

Selection of this parameter, of type Identifier, indicates that all Event Condition object whose scope is the Domain whose name is provided by this parameter are candidates for deletion.

19.3.1.1.1.4 VMD Specific

Selection of this parameter shall indicate that all Event Condition objects whose scope is the VMD are candidates for deletion.

19.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

19.3.1.2.1 Candidates Not Deleted

This parameter, of type integer, shall contain the number of Event Condition objects that were included in the scope of Event Condition objects to be deleted, but that were not deleted because of a non-empty value of the &EventEnrollments field, or because deletion was not permitted.

19.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

19.3.2 Service Procedure

19.3.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

19.3.2.2 Actions

The MMS server shall prepare a list of objects to be deleted as indicated by the Scope of Delete parameter. It shall initialize the Candidates Not Deleted parameter to zero.

For each object on the list, the MMS server shall perform the following checks:

a) The Event Condition object exists;

b) The conditions in the Access Control List specified by the &accessControl field of the object to be deleted shall be satisfied for service class = DELETE;

c) The &EventEnrollments field shall be emtpy;

d) The &ReferencingEventConditionLists field (if present) shall be empty.

If any of these conditions is not met, the MMS server shall increment by one the Candidates Not Deleted value response parameter and proceed to the next Event Condition object. Otherwise, the MMS server shall remove the reference to this Event Condition object from the &EventConditions field of the Access Control List object referenced by the &accessControl field of this Event Condition object and delete the Event Condition object.

After the complete list has been processed, the MMS server shall return a Result(+) with the Candidates Not Deleted parameter set to the number of Event Condition objects that were not deleted.

NOTE If this count is not equal to zero, the requesting MMS-user (the client) may use the GetNameList service to determine the Event Condition objects that were not deleted.

19.4 GetEventConditionAttributes service

The GetEventConditionAttributes service is used by an MMS client to obtain the attributes of an Event Condition object at a VMD.

19.4.1 Structure

The structure of the component service primitives is shown in Table 108.

Table 108 - GetEventConditionAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Condition Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
Class			M	M(=)	
Priority			M	M(=)	
Severity			M	M(=)	
Alarm Summary Reports			C	C(=)	
Monitored Variable			C	C(=)	
Evaluation Interval			C	C(=)	
Access Control List			C	C(=)	aco
Group Priority Override			C	C(=)	cspi
List of Referencing ECL			C	C(=)	cspi
Display Enhancement			C	C(=)	cspi
String			S	S(=)	
Index			S	S(=)	
No Enhancement			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

19.4.1.1 Argument

This parameter shall convey the parameter of the GetEventConditionAttributes service request.

19.4.1.1.1 Event Condition Name

This parameter, of type Object Name, shall contain the name of the Event Condition object for which the attributes are to be obtained.

19.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

19.4.1.2.1 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Event Condition object may be deleted using the DeleteEventCondition service. Subclause 9.1.4 specifies the value to be returned by this parameter.

19.4.1.2.2 Class

This parameter, of type EC-Class, shall contain the value of the Event Condition object's &ecClass field.

19.4.1.2.3 Priority

This parameter, of type Priority, shall contain the current value of the Event Condition object's &priority field.

19.4.1.2.4 Severity

This parameter, of type integer, shall contain the value of the Event Condition object's &severity field.

19.4.1.2.5 Alarm Summary Reports

For a **monitored** Event Condition object, this parameter, of type boolean, shall contain the value of the Event Condition object's &alarmSummaryReports field. This parameter shall be omitted for a **network-triggered** Event Condition object.

19.4.1.2.6 Monitored Variable

For a **monitored** Event Condition object, the Monitored Variable parameter, of type Variable Specification, shall contain the value of the key attribute (the &name attribute for a Named Variable, or the &address field for an Unnamed Variable) from the object referenced by the Event Condition object's &monitoredVariable field. If the value of the &monitoredVariable field is **unspecified**, this parameter shall be omitted. If the object referenced by the &monitoredVariable field becomes unavailable, the Monitored Variable parameter shall have the value UNDEFINED.

19.4.1.2.7 Evaluation Interval

For a **monitored** Event Condition object, this parameter, of type integer, shall contain the value of the &evaluationInterval field of the Event Condition object. This parameter shall be omitted for a **network-triggered** Event Condition object.

19.4.1.2.8 Access Control List

This parameter, of type Identifier, shall indicate the name of the Access Control List object that controls access to this Event Condition object. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

19.4.1.2.9 Group priority override

This parameter shall contain the value of the &group-Priority-Override field of the Event Condition object. This parameter shall not appear unless the **cspi** parameter CBB has been negotiated.

NOTE This parameter need not appear if Event Condition Lists are not supported.

19.4.1.2.10 List of Referencing ECL

This parameter shall contain a list of names, derived from the contents of the &ReferencingEventConditionLists field of the Event Condition object. Each name shall be equal to the value of the &name field of a referencing Event Condition List object. This parameter shall not appear unless the **cspi** parameter CBB has been negotiated.

NOTE This parameter need not appear if Event Condition Lists are not supported.

19.4.1.2.11 Display Enhancement

This parameter shall indicate the value of the &displayEnhancement field of the Event Condition object. This parameter shall not appear unless the **cspi** parameter CBB has been negotiated. Depending on its value, one of the following parameters shall be selected.

19.4.1.2.11.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter.

19.4.1.2.11.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter.

19.4.1.2.11.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present.

19.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

19.4.2 Service Procedure

The MMS server issue a Result(+) containing the values of the specified parameters.

19.5 ReportEventConditionStatus service

The ReportEventConditionStatus service is used by an MMS client to obtain the status of an Event Condition object from the MMS server.

19.5.1 Structure

The structure of the component service primitives is shown in Table 109.

Table 109 - ReportEventConditionStatus service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Condition Name	M	M(=)			
Result(+)			S	S(=)	
Current State			M	M(=)	
Number of Event Enrollments			M	M(=)	
Enabled			C	C(=)	
Time Of Last Transition to Active			C	C(=)	
Time Of Last Transition to Idle			C	C(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

19.5.1.1 Argument

This parameter shall convey the parameter of the ReportEventConditionStatus service request.

19.5.1.1.1 Event Condition Name

This parameter, of type Object Name, shall contain the &name field of the Event Condition object for which the status report is requested.

19.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

19.5.1.2.1 Current State

This parameter, of type EC-State, shall contain the value of the Event Condition object's &ecState field.

19.5.1.2.2 Number Of Event Enrollments

This parameter, of type integer, shall contain the number of entries in the Event Condition object's &EventEnrollments field.

19.5.1.2.3 Enabled

This parameter, of type boolean, shall contain the value of the &enabled field of the Event Condition object for a **monitored** Event Condition object. This parameter shall be omitted for a **network-triggered** Event Condition object.

19.5.1.2.4 Time Of Last Transition To Active

For a **monitored** Event Condition object having a &timeToActive field with value not equal to **undefined**, this parameter, expressed as a date and time of day or a Time Sequence Identifier, shall contain the current value of the &timeToActive field. Otherwise, this parameter shall be omitted.

19.5.1.2.5 Time Of Last Transition To Idle

For a **monitored** Event Condition object having a &timeToIdle field with value not equal to **undefined**, this parameter, expressed as a date and time of day or a Time Sequence Identifier, shall contain the current value of the &timeToIdle field. Otherwise, this parameter shall be omitted.

19.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

19.5.2 Service Procedure

The MMS server shall issue a Result(+) containing the values of the specified parameters.

19.6 AlterEventConditionMonitoring service

The AlterEventConditionMonitoring service is used by an MMS client to request that the MMS server alter any combination of a monitored Event Condition object's &enable, &priority, &alarmSummaryReports and &evaluationInterval fields.

19.6.1 Structure

The structure of the component service primitives is shown in Table 110.

Table 110 - AlterEventConditionMonitoring service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Condition Name	M	M(=)			
Enabled	U	U(=)			
Priority	U	U(=)			
Alarm Summary Reports	U	U(=)			
Evaluation Interval	U	U(=)			cei
Display Enhancement	U	U(=)			cspi
String	S	S(=)			des
Index	S	S(=)			dei
No Enhancement	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

19.6.1.1 Argument

This parameter shall convey the parameters of the AlterEventConditionMonitoring service request. At least one of Enabled, Priority, Alarm Summary Reports, or Evaluation Interval shall be present.

19.6.1.1.1 Event Condition Name

This parameter, of type Object Name, shall specify the name of the Event Condition object to be altered.

19.6.1.1.2 Enabled

This parameter, of type boolean, shall specify (if included) the desired replacement value for the Event Condition object's &enabled field.

If the Enabled parameter is not specified, the &enabled field shall not be changed.

19.6.1.1.3 Priority

This parameter, of type Priority, shall specify (if included) the desired replacement value for the Event Condition object's &priority field.

If the Priority parameter is omitted, the &priority field of the Event Condition object shall not be changed.

NOTE Depending on the particular use of priority by an implementation, it is possible that assignment of increased priority may result in a period during which EventNotification requests primitives are not sequentially ordered with respect to time of occurrence.

19.6.1.1.4 Alarm Summary Reports

This parameter, of type boolean, shall specify (if included) the desired replacement value for the Event Condition object's &alarmSummaryReports field.

If the Alarm Summary Reports parameter is not specified, the value of the &alarmSummaryReports field shall not be changed.

19.6.1.1.5 Evaluation Interval

This parameter, of type integer, shall specify (if included) a new proposed value for the Evaluation Interval attribute of the Event Condition object. This parameter shall not be included unless the **cei** CBB has been negotiated.

This parameter, if included, shall be provided as guidance to the MMS server. Acceptance of this value shall guarantee only that the MMS server shall make every effort to honour the specified value. A MMS server that determines itself, through local means, to be unable to honour the service request, due to resource or other limitations, shall return a TIME-RESOLUTION error in response to this service request.

19.6.1.1.5.1 Display Enhancement

This parameter shall specify (if included) that the &displayEnhancement field of the Event Condition shall be altered by this service. This parameter shall not appear unless **cspi** CBB has been negotiated. If this parameter is included, one of the following parameters shall be selected.

19.6.1.1.5.1.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the **des** CBB has been negotiated.

19.6.1.1.5.1.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the **dei** CBB has been negotiated.

19.6.1.1.5.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither **des** nor **dei** has been negotiated.

19.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

19.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

19.6.2 Service Procedure

19.6.2.1 Preconditions

The MMS server shall verify that:

a) the Event Condition object identified by the Event Condition Name parameter is a **monitored** Event Condition.

b) all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

c) that all the conditions in the Access Control List object referenced by the &accessControl field of the Event Condition object are satisfied for the service class = LOAD.

If any of these conditions is not satisfied, the service shall fail and a Result(-) shall be returned.

19.6.2.2 Actions

The monitored Event Condition object specified by the Event Condition Name parameter shall have one or more of its attribute field values altered.

a) If the Enabled parameter is present in the service request, the &enabled field shall be altered.

b) If the Priority parameter is present in the service request, the &priority field shall be altered.

c) If the Alarm Summary Reports parameter is present in the service request, the &alarmSummaryReports field shall be altered.

d) If the **cei** CBB has been negotiated and if the Evaluation Interval parameter is present in the service request, the &evaluationInterval field shall be altered.

e) If the **cspi** CBB has been negotiated and if the Display Enhancement parameter is present in the service request, the &displayEnhancement field shall be altered. If the String choice is selected, the &displayEnhancement field shall be set to the value of the String parameter. If the Index choice is selected, the &displayEnhancement field shall be set to the value of the Index parameter. Otherwise, the &displayEnhancement field shall be set to **none**.

A Result(+) response shall be issued.

After the response is issued, if an included value for the Enabled parameter caused a change in the value of the Event Condition object's &enabled field, the Procedure for Event Transition Processing (18.1.1) shall be executed for the indicated transition.

20 Event Action services

20.1 Event Actions

This clause provides an object model for the following object:

EVENT-ACTION

This clause specifies the following services:

DefineEventAction	GetEventActionAttributes
DeleteEventAction	ReportEventActionStatus

An Event Action object is an MMS confirmed service that shall be executed whenever a specified transition of an Event Condition object's &ecState field is detected. An Event Action object represents a component of the Procedure for Event Transition Processing (see 18.1.1). The attributes of an Event Action object are described below, followed by a description of the services that operate on the Event Action object.

20.1.1 The Event Action object

This clause introduces the model of the Event Action Object.

```
EVENT-ACTION ::= CLASS {
    &name                       ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
    &accessControl              ACCESS-CONTROL-LIST,
    &confirmedServiceRequest    ConfirmedServiceRequest,
    &Modifiers                  Modifier,
    &EventEnrollments           EVENT-ENROLLMENT }
```


20.1.1.1 &name

The &name field uniquely identifies the Event Action object within the VMD. It shall have VMD-specific, Domain-specific or AA-specific scope.

20.1.1.2 &accessControl

The &accessControl field identifies an Access Control List object that provides conditions under which this Event Action object may be deleted or have its access control changed.

20.1.1.3 &confirmedServiceRequest

The &confirmedServiceRequest field identifies the service procedure to be executed by the MMS server as part of the Procedure for Event Transition Processing. This field contains the service argument to be used when executing this service procedure.

20.1.1.4 &Modifiers

The &Modifiers field specifies an ordered set of zero or more modifiers that apply to the execution of the Event Action.

20.1.1.5 &EventEnrollments

The &EventEnrollments field identifies zero or more Event Enrollment objects each of which refers to this Event Action object.

20.2 DefineEventAction service

The DefineEventAction service is used by an MMS client to request the creation of an Event Action object at a VMD.

20.2.1 Structure

The structure of the component service primitives is shown in Table 111.

Table 111 - DefineEventAction service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Action Name	M	M(=)			
List of Modifier	U	U(=)			
Confirmed Service Request	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

20.2.1.1 Argument

This parameter shall convey the parameters of the DefineEventAction service request.

20.2.1.1.1 Event Action Name

This parameter, of type Object Name, shall contain the value to be assigned to the &name field of the Event Action object to be created.

20.2.1.1.2 List Of Modifier

This parameter shall contain the modifiers, if any, that apply for each execution of this Event Action (see 5.6).

NOTE This list does not apply to the execution of the DefineEventAction service.

20.2.1.1.3 Confirmed Service Request

This parameter shall contain a choice of a confirmed service and a valid argument, as specified by the Argument parameter of the request primitive of that confirmed service. The responder role for this service shall have been included in the list of supported services negotiated by the MMS server, through the use of the Initiate service, when the current Application Association was established.

20.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

20.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

20.2.2 Service Procedure

20.2.2.1 Preconditions

The MMS server shall verify that no Event Action object with the same name as the Event Action Name parameter. If this condition is not satisfied, a Result(-) shall be returned.

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD. If this condition is not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED.

20.2.2.2 Actions

An Event Action object shall be created and initialized as follows:

a) &name - Initialized to the value of the Event Action Name parameter.

b) &accessControl - Initialized to refer to an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) &Modifiers - Initialized to the value of the List Of Modifier parameter, if present. Otherwise initialized to empty.

d) &confirmedServiceRequest - Initialized to the value of the Confirmed Service Request parameter.

e) &EventEnrollments - Initialized to empty.

A Result(+) shall be returned, indicating creation of the Event Action object.

20.3 DeleteEventAction service

The DeleteEventAction service is used by an MMS client to request that an MMS server delete one or more Event Action objects.

20.3.1 Structure

The structure of the component service primitives is shown in Table 112.

Table 112 - DeleteEventAction service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
Event Action Names	S	S(=)			
AA Specific	S	S(=)			
Domain Name	S	S(=)			
VMD Specific	S	S(=)			
Result(+)			S	S(=)	
Candidates Not Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

20.3.1.1 Argument

This parameter shall convey the parameters of the DeleteEventAction service request.

20.3.1.1.1 Scope Of Delete

The Scope Of Delete parameter shall specify the extent of delete to be attempted. One of the following parameters shall be selected.

20.3.1.1.1.1 Event Action Names

Selection of this parameter, containing one or more Event Action Names each of type Object Name and each identifying an Event Action object, provides the names of the specific candidate Event Action objects as candidates for deletion.

20.3.1.1.1.2 AA Specific

Selection of this parameter indicates that all Event Action objects whose scope is the current application association are candidates for deletion.

20.3.1.1.2 Domain Name

Selection of this parameter, of type Identifier, indicates that all Event Action objects whose scope is the specified Domain are candidates for deletion.

20.3.1.1.2.1 VMD Specific

Selection of this parameter indicates that all Event Action objects whose scope is VMD are candidates for deletion.

20.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

20.3.1.2.1 Candidates Not Deleted

This parameter, of type integer, shall contain the number of Event Action objects that were candidates for deletion but were not deleted because of a non-empty value of the &EventEnrollment field, or because the conditions in the Access Control List referenced by the &accessControl field of the Event Action object were not satisfied for Service Class = DELETE.

20.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

20.3.2 Service Procedure

20.3.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE. If this condition is not satisfied, the service request fails and a Result(-) shall be returned.

20.3.2.2 Actions

The MMS server shall prepare a list of objects to be deleted as indicated by the Scope of Delete parameter. It shall initialize the Candidates Not Deleted parameter to zero.

For each object on the list, the MMS server shall perform the following checks:

a) The Event Action object exists;

b) The conditions in the Access Control List specified by the &accessControl field of the object to be deleted shall be satisfied;

c) The &EventEnrollments field shall be emtpy;

If any of these conditions is not met, the MMS server shall increment by one the Candidates Not Deleted value response parameter and proceed to the next Event Action object. Otherwise, the MMS server shall remove the reference to this Event Action object from the &EventActions field of the Access Control List object referenced by the &accessControl field of this Event Action object and delete the Event Action object.

After the complete list has been processed, the MMS server shall return a Result(+) with the Candidates Not Deleted parameter set to the number of Event Action objects that were not deleted.

NOTE If this count is not equal to zero, the requesting MMS-user (the client) may use the GetNameList service to determine the Event Action objects that were not deleted.

20.4 GetEventActionAttributes service

The GetEventActionAttributes service is used by an MMS client to obtain from an MMS server the values of the attributes of an Event Action object at the VMD.

20.4.1 Structure

The structure of the component service primitives is shown in Table 113.

Table 113 - GetEventActionAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Action Name	M	M(=)			
Result(+)			S	S(=)	
MMS Deletable			M	M(=)	
List Of Modifier			M	M(=)	
Confirmed Service Request			M	M(=)	
Access Control List			C	C(=)	aco
Result(-)			S	S(=)	
Error Type			M	M(=)	

20.4.1.1 Argument

This parameter shall convey the parameters of the GetEventActionAttributes service request.

20.4.1.1.1 Event Action Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Action object for which the attributes are requested.

20.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

20.4.1.2.1 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) this Event Action object may be deleted using the DeleteEventAction service. Subclause 9.1.4 specifies the value to be returned by this parameter.

20.4.1.2.2 List Of Modifier

This parameter shall contain the value of the &Modifiers field of the Event Action object.

20.4.1.2.3 Confirmed Service Request

This parameter shall contain the value of the Event Action object's &confirmedServiceRequest field.

20.4.1.2.4 Access Control List

This parameter, of type Identifier, shall indicate the &name field of the Access Control List object referenced by the &accessControl field of this Event Action object. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

20.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

20.4.2 Service Procedure

The MMS server shall issue a Result(+) containing the value of the specified parameters.

20.5 ReportEventActionStatus service

The ReportEventActionStatus service is used by an MMS client to obtain a count of the number of Event Enrollment objects that are currently specifying an Event Action object.

20.5.1 Structure

The structure of the component primitives is shown in Table 114.

Table 114 - ReportEventActionStatus service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Action Name	M	M(=)			
Result(+)			S	S(=)	
Number of Event Enrollments			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

20.5.1.1 Argument

This parameter shall convey the parameter of the ReportEventActionStatus service request.

20.5.1.1.1 Event Action Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Action object for which status is desired.

20.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

20.5.1.2.1 Number Of Event Enrollments

This parameter, of type integer, shall contain the count of Event Enrollment objects referenced by the &EventEnrollments field of the Event Action object.

20.5.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

20.5.2 Service Procedure

20.5.2.1 Preconditions

If no Event Action object whose &name field matches the Event Action Name parameter, a Result(-) shall be returned.

20.5.2.2 Actions

The MMS server shall issue a Result(+) containing the count of entries in the Event Action object's &EventEnrollments field.

20.5.3 The Event Action State Diagram

The state diagram for an event action is shown in Figure 20.

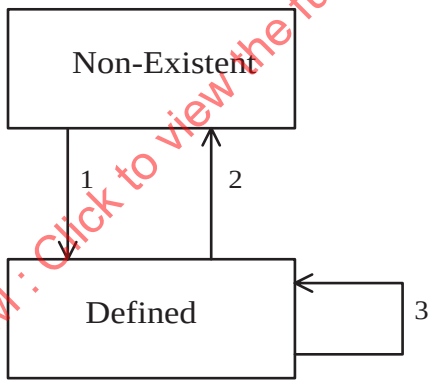

Figure 20 - Event Action State Diagram

Transitions:

1. Receive DefineEventAction indication.

2. Receive DeleteEventAction indication with the conditions for deletion of an Event Action object being satisfied (see 20.3.2), or loss of application association if the Event Action object is of AA-specific scope and is dependent on the lost application association, or deletion of a Domain if the Event Action object is of Domain-specific scope and dependent on the deleted Domain.

3. Receipt of any of the following indications:

- DeleteEventAction with the conditions for deletion of an Event Condition object being satisfied (see 20.3.2)
- GetEventActionAttributes;
- ReportEventActionStatus;
- DefineEventEnrollment;
- DeleteEventEnrollment; or
- GetEventEnrollmentAttributes;

21 Event Enrollment services

21.1 Event Enrollments

This clause provides an object model for the following object:

EVENT-ENROLLMENT

This clause specifies the following services:

DefineEventEnrollment
AlterEventEnrollment
DeleteEventEnrollment
GetEventEnrollmentAttributes
ReportEventEnrollmentStatus

An Event Enrollment represents a request from an MMS user to be notified of the occurrence of one or more specified transitions of an Event Condition object as reflected in its &ecState field, or to delay execution of a confirmed MMS-service until the occurrence of one or more specified transitions of an Event Condition object. An MMS user receiving a notification of a transition as a result of an Event Enrollment for the specified transition, may acknowledge receipt of the transition notification to the MMS user issuing the notification.

21.1.1 The Event Enrollment object

This clause introduces the model of the Event Enrollment Object

```
EVENT-ENROLLMENT ::= CLASS {
    &name                       ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
    &accessControl              ACCESS-CONTROL-LIST,
    &eeClass                    EE-Class ,
    &eventCondition             EVENT-CONDITION,
    &ecTransitions              Transitions,
    &aAssociation               APPLICATION-ASSOCIATION,
-- The following two fields are present if and only if the
-- value of &eeClass is modifier.
    &invokeID                   INTEGER OPTIONAL,
    &remainingDelay             CHOICE {
        time                        INTEGER,
        forever                     NULL } OPTIONAL,
-- All the following fields are present if and only if the
-- value of &eeClass is notification.
    &notificationLost           BOOLEAN OPTIONAL,
    &eventAction                EVENT-ACTION OPTIONAL,
    &duration                   EE-Duration OPTIONAL,
    &clientApplication          ApplicationReference OPTIONAL,
-- The following four fields are present if and only if the
-- value of &eeClass is notification and the value of &ecState
-- of the Event Condition object is monitored
    &aaRule                     AlarmAckRule OPTIONAL,
    &timeActiveAck              EventTime OPTIONAL,
    &timeIdleAck                EventTime OPTIONAL,
    &ackState                   ENUMERATED {
        acked                       (0),
        noAckA                      (1),
        noAckI                      (2) } OPTIONAL,
    &lastState                  EC-State OPTIONAL,
IF (cspi)
    &displayEnhancement         CHOICE {
IF (des)
        text                        MMSString,
ENDIF
```

```
IF (dei)
        number                  INTEGER,
ENDIF
        none                    NULL } OPTIONAL
ENDIF
    }
```

21.1.1.1 &name

The &name field uniquely identifies the Event Enrollment object within the VMD. It may be of VMD-specific, Domain-specific or AA-specific scope.

21.1.1.2 &accessControl

The &accessControl field specifies the Access Control List object that provides conditions under which this Event Enrollment object may have its attributes changed, may be deleted, or may have its access control changed.

21.1.1.3 &eeClass

The &eeClass field indicates the class of the Event Enrollment object. Two values for the &eeClass field are defined:

```
EE-Class ::= INTEGER {
    modifier        (0),
    notification    (1) }
```

21.1.1.3.1 modifier

The Event Enrollment object represents a temporary (executed one time only and deleted) Event Enrollment, created as a result of receipt of a service indication specifying a confirmed MMS service that was modified by the Attach To Event Condition service modifier. An Event Enrollment object having the value of the &eeClass field equal to **modifier** is referred to as a modifier Event Enrollment object. Modifier Event Enrollment objects are not alterable through the AlterEventEnrollment service or deletable through the DeleteEventEnrollment service.

21.1.1.3.2 notification

The Event Enrollment object is an explicitly defined or predefined Event Enrollment object. It represents a request that the Procedure for Event Transition Processing (see 18.1.1) be executed upon detection of any of the indicated Event Condition transitions. An Event Enrollment object having the value of the &eeClass field equal to **notification** is referred to as a **notification** Event Enrollment object.

21.1.1.4 &eventCondition

The &eventCondition field identifies the Event Condition object that will cause invocation of the Procedure for Event Transition Processing (see 18.1.1) for this Event Enrollment object. If the referenced Event Condition object becomes unavailable for use, for example through the deletion of a Domain or termination of an application association, the value of this field becomes undefined.

21.1.1.5 &ecTransitions

If the Event Enrollment object references a **monitored** Event Condition, the &ecTransitions field contains the set of Event Condition transitions that cause the invocation of the Procedure for Event Transition Processing for this Event Enrollment object. It consists of any non-empty set composed of members chosen from the elements **disabled-to-active**, **disabled-to-idle**, **idle-to-active**, **idle-to-disabled**, **active-to-idle**, **active-to-disabled**, and **any-to-deleted**. For a notification Event Enrollment object, if the value of the Event Enrollment object's &aaRule field is not equal to **none**, this field shall include transitions to the **active** state. If the value of the &aaRule is **ack-all**, this field shall also include transitions to the **idle** state.

If the Event Enrollment object references a **network-triggered** Event Condition object, this field specifies the empty set and the Procedure for Event Transition Processing for this Event Enrollment shall be executed only if the event is explicitly triggered by

an MMS-client using the TriggerEvent service or if the **network-triggered** event occurs as a result of an autonomous action of the MMS server.

If the &ecTransitions field contains the element **any-to-deleted**, the Procedure for Event Transition Processing shall be performed if the Event Condition object referenced by the &eventCondition field of the Event Enrollment object becomes undefined, or if the &monitoredVariable field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object changes to **unspecified**.

```
Transitions ::= BIT STRING {
    idle-to-disabled        (0),
    active-to-disabled      (1),
    disabled-to-idle        (2),
    active-to-idle          (3),
    disabled-to-active      (4),
    idle-to-active          (5),
    any-to-deleted          (6) }
```

21.1.1.6 &aAssociation

The &aAssociation field identifies the application association that shall be used for EventNotifications in the case of notification Event Enrollment objects.

This attribute cannot be reported through any of the MMS services. The &abstractSyntax field of this application-association shall be:

a) the abstract syntax in use on the application association over which the DefineEventEnrollment service request or AttachToEventCondition modifier was received, or

b) as specified locally if the Event Enrollment object was not created through the use of an MMS service.

21.1.1.7 &invokeID

The &invokeID field exists only for **modifier** Event Enrollment objects. It shall contain the &invokeID field of the Transaction object of the modified service.

21.1.1.8 &remainingDelay

The &remainingDelay field exists only for **modifier** Event Enrollment objects. It shall contain either the value of the time in seconds for which the MMS-user requesting the modifier Event Enrollment object is willing to wait for the specified Event Condition transitions to occur or the value **forever**. If this field has the value **forever**, the acceptable delay is infinite.

21.1.1.9 &notificationLost

The &notificationLost field exists only for **notification** Event Enrollment objects. The value of this field shall be true if completion of the Procedure for Event Transition Processing has been inhibited due to resource limitations at the MMS server, or if a notification cannot be sent due to the inability of the MMS server to establish an application association for Event Enrollment objects having a value for the &duration field of **permanent**. Otherwise, the value of the &notificationLost field shall be false.

21.1.1.10 &eventAction

The &eventAction field exists only for **notification** Event Enrollment objects. Its value may be undefined, indicating that the Event Enrollment requests only an EventNotification, or it may refer to an Event Action object, indicating that the Event Enrollment requests an EventNotification containing the result of execution of the specified Event Action. In the case that the Event Action object referenced by the &eventAction field becomes unavailable, for example through the deletion of a Domain (if the Event Action object is Domain specific) or termination of an application association (if the Event Action object is AA-specific), the value of this field shall become undefined.

21.1.1.11 &duration

The &duration field exists only for **notification** Event Enrollment objects. Its value shall indicate the duration of the Event Enrollment object and may have either of two values:

current - indicates that the Event Enrollment object is defined for the life of the application association over which the Event Enrollment object was defined.

permanent - indicates that the Event Enrollment object is defined for the life of the VMD unless explicitly deleted.

```
EE-Duration ::= INTEGER {
    current         (0),
    permanent       (1) }
```

21.1.1.12 &clientApplication

The &clientApplication field exists only for **notification** Event Enrollment objects. It contains the identification of the enrolled client application and is of type ApplicationReference.

21.1.1.13 &aaRule

NOTE In the following subclause "required" acknowledgements are acknowledgements that are required by this part of ISO 9506. "Allowed" acknowledgements are acknowledgements that may, at the discretion of the MMS-user issuing the acknowledgement, be transmitted to the MMS-user issuing the Event Notification, but they have the effect only of updating the &timeActiveAck field, or &timeIdleAck field, as appropriate, of the specified Event Enrollment object.

```
AlarmAckRule ::= INTEGER {
    none            (0),
    simple          (1),
    ack-active      (2),
    ack-all         (3) }
```

The &aaRule field exists only for a **notification** Event Enrollment object that reference a **monitored** Event Condition object. It indicates the level of acknowledgement required for EventNotification service instances generated as a result of the Event Enrollment.

The value of this attribute is also considered in determining whether or not the Event Enrollment object will be included in alarm enrollment summaries (see 18.6).

This field contains one of the following values:

21.1.1.13.1 none

Acknowledgements to an Event Notification generated by this Event Enrollment object are allowed but not required. If an acknowledgement to an Event Notification is received, it has no effect on the &ackState field of the Event Enrollment object.

21.1.1.13.2 simple

Acknowledgements to an Event Notification generated by this Event Enrollment object are allowed but not required. If an acknowledgement to an Event Notification reporting a transition to the **active** state is received, it causes a change in the &ackState field of the Event Enrollment object. Transitions to other states have no effect on the &ackState field of the Event Enrollment object.

21.1.1.13.3 ack-active

Acknowledgements to an Event Notification generated by this Event Enrollment object reporting a transition to the **active** state are required. Acknowledgements to an Event Notification generated by this Event Enrollment object reporting a transition to a state other than the **active** state are allowed but not required. If an acknowledgement to an Event Notification reporting a transition

to the **active** state is received, it causes a change in the &ackState field of the Event Enrollment object. Transitions to other states have no effect on the &ackState field of the Event Enrollment object.

21.1.1.13.4 ack-all

Acknowledgements to an Event Notification generated by this Event Enrollment object are required. If an acknowledgement to an Event Notification reporting a transition to the **active** state or to the **idle** is received, it causes a change in the &ackState field of the Event Enrollment object. Transitions to other states have no effect on the &ackState field of the Event Enrollment object.

NOTE Acknowledgement are never be required for an event notification specifying a transition into or out of the **disabled** state.

21.1.1.14 &timeActiveAck

The &timeActiveAck field exists only for **notification** Event Enrollment objects that reference a **monitored** Event Condition object. The value of the &timeActiveAck field is only meaningful if the value of the &aaRule field is not equal to **none**. The &timeActiveAck field records the time (date and time of day or Time Sequence Identifier) at which an acknowledgement for the most recently detected transition of the Event Condition object to the **active** state was received from the enrolled client. If this acknowledgement has not been received, the value of the &timeActiveAck field is **undefined**.

21.1.1.15 &timeIdleAck

The &timeIdleAck field exists only for **notification** Event Enrollment objects that reference a **monitored** Event Condition object. The value of the &timeIdleAck field is only meaningful if the value of the &aaRule field is not equal to **none**. The &timeIdleAck field records the time (date and time of day or Time Sequence Identifier) at which an acknowledgement for the most recently detected transition of the Event Condition object to the **idle** state was received from the enrolled client. If this acknowledgement has not been received, the value of the &timeIdleAck field is **undefined**.

21.1.1.16 &ackState

The &ackState field indicates the state of the Event Enrollment object with respect to acknowledgements. The possible values of the &state field depend on the value of the &aaRule field.

21.1.1.16.1 acked

All required acknowledgements of event notifications have been received.

21.1.1.16.2 noAckA

The &timeActiveAck field of this Event Enrollment object is either **undefined** or has a value earlier than the &timeToActive field of the Event Condition referenced by the &eventCondition field of the Event Enrollment object.

21.1.1.16.3 noAckI

The &timeIdleAck field of this Event Enrollment object is either **undefined** or has a value earlier than the &timeToIdle field of the Event Condition referenced by the &eventCondition field of the Event Enrollment object.

21.1.1.17 &lastState

The &lastState field exists only for **notification** Event Enrollment objects. If the Event Condition object referenced by the &eventCondition field of the Event Enrollment has become unavailable, either through the deletion of a Domain or the loss of an application association, the &lastState field reflects the last known state of the Event Condition object. Otherwise, the meaning of this field is undefined.

21.1.1.18 &displayEnhancement

This field specifies the type of the &displayEnhancement field of the Event Enrollment object. This field is present only if the **cspi** parameter CBB has been negotiated. If the value of this field is **text**, the &displayEnhancement field is of type character string. If the value of this field is **number**, the &displayEnhancement is of type integer. If the value of this field is **none**, the &displayEnhancement field is null.

21.1.2 The Event Enrollment State diagrams

The state diagrams of Figure 21 to Figure 24 describe the &ackState of an Event Enrollment. The arcs shown in the various Event Enrollment state diagrams are defined as follows.

1. Receive AlterEventConditionMonitoring indication specifying the Enabled parameter equal to true while the variable referenced by the Event Condition's &monitoredVariable field is true, or a locally defined event is pending.

2. Receive AlterEventConditionMonitoring indication specifying the Enabled parameter equal to true while the variable referenced by the Event Condition's &monitoredVariable field is false, or a locally defined event is not pending.

3. The variable referenced by the Event Condition's &monitoredVariable field becomes true or a locally defined event is detected.

4. The variable referenced by the Event Condition's &monitoredVariable field becomes false or the condition causing a locally defined event is cleared.

5. Receive AcknowledgeEventNotification indication for current state and time of transition.

6. Receive AcknowledgeEventNotification indication for **active** state while in the **idle** state and time is equal to Event Condition's &timeToActive field.

7. Receive AlterEventConditionMonitoring indication specifying the Enabled parameter equal to false.

Additional arcs are possible if an AlterEventEnrollment indication is processed that changes the value of the &aaRule field. In this case the state diagram regulating the event enrollment is also changed.

Receipt of an indication for any of the following services does not change the state of an Event Enrollment.

GetEventEnrollmentAttributes	GetAlarmEnrollmentSummary
GetAlarmSummary	ReportEventEnrollmentStatus

Arcs indicating transitions that do not cause a state change are not shown.

Receipt of a DefineEventEnrollment indication establishes a new Event Enrollment. This is modeled as a transition from a non-existent state. Receipt of a DeleteEventEnrollment indication is modeled as a transition from the current state to the non-existent state. Arcs indicating transitions to or from the non-existent state are not shown.

21.1.2.1 &aaRule equals none

Figure 21 contains the state diagram for an Event Enrollment specifying the &aaRule field equal to **none**.

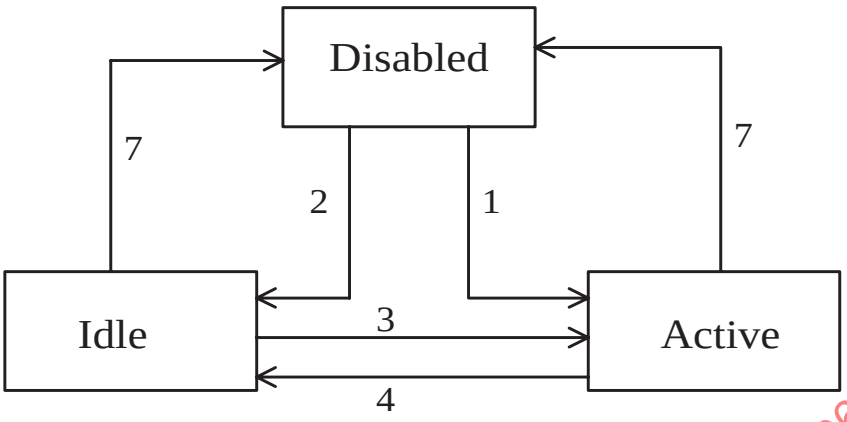

Figure 21 - State Diagram for &alarmAcknowledgmentRule = none

21.1.2.2 &aaRule equals simple

Figure 22 contains the state diagram for an Event Enrollment specifying the &aaRule field equal to **simple**.

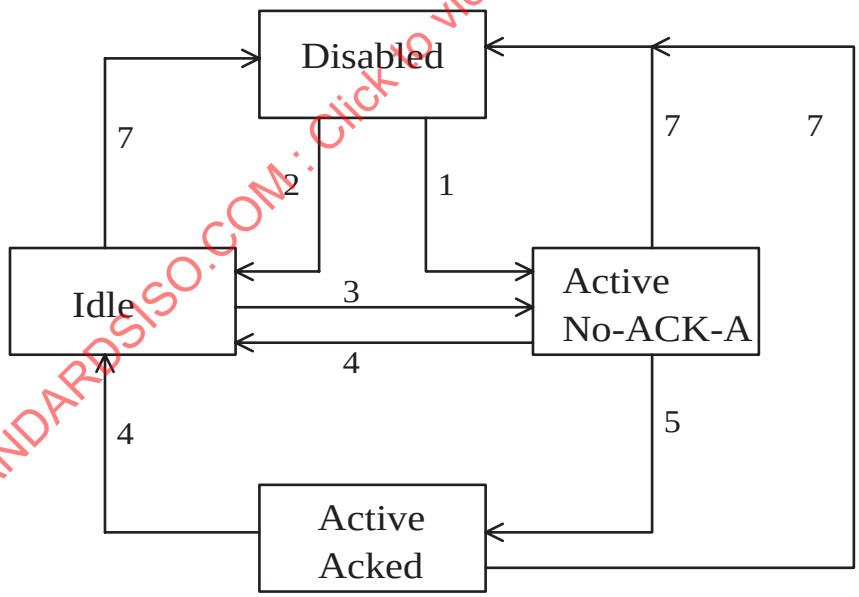

Figure 22 - State Diagram for &alarmAcknowledgmentRule = simple

21.1.2.3 &aaRule equals ack-active

Figure 23 contains the state diagram for an Event Enrollment specifying the &aaRule field equal to ack-active.

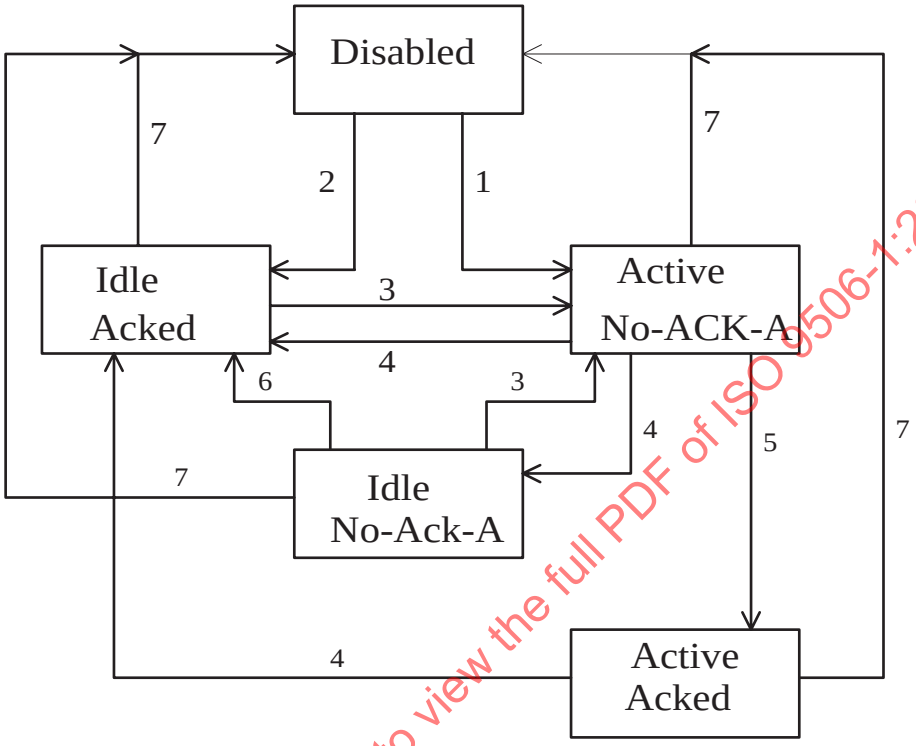

Figure 23 - State Diagram for &alarmAcknowledgmentRule = ack-active

21.1.2.4 &aaRule equals ack-all

Figure 24 contains the state diagram for an Event Enrollment specifying the &aaRule field equal to ack-all.

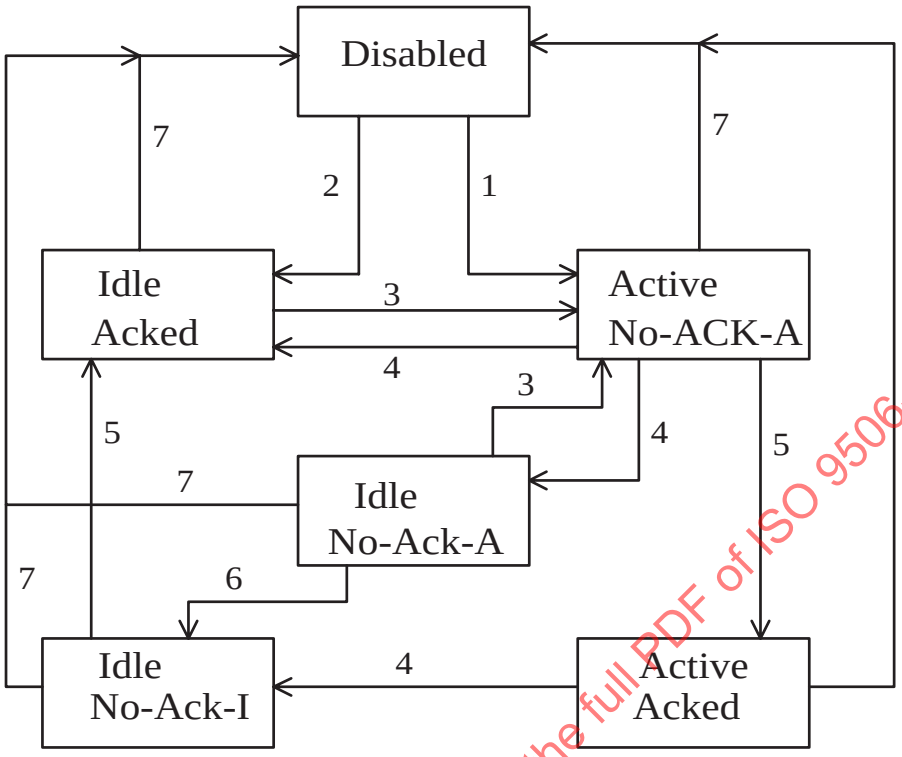

Figure 24 - State Diagram for &alarmAcknowledgmentRule = ack-all

21.2 DefineEventEnrollment service

The DefineEventEnrollment service is used by an MMS client to request the creation of an Event Enrollment object at a VMD. This service has the effect that the MMS server add the requesting MMS client or another "third party" client to the list of users for which the Procedure for Event Transition Processing (see 18.1.1) is to be executed as a result of a specified transition or set of transitions of an Event Condition object.

21.2.1 Structure

The structure of the component service primitives is shown in Table 115.

Table 115 - DefineEventEnrollment service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Enrollment Name	M	M(=)			
Event Condition Name	M	M(=)			
Event Condition Transactions	M	M(=)			
Alarm Acknowledgement Rule	M	M(=)			
Event Action Name	U	U(=)			
Client Application	U	U(=)			tpy
Display Enhancement	M	M(=)			cspi
String	S	S(=)			des
Index	S	S(=)			dei
No Enhancement	S	S(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	
Object Not Defined			C	C(=)	

21.2.1.1 Argument

This parameter shall convey the parameters of the DefineEventEnrollment service request.

21.2.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall be the value of the &name field of the newly created Event Enrollment object. This name shall be unique among all Event Enrollment objects with identical scope (VMD-specific, Domain-specific or AA-Specific).

21.2.1.1.2 Event Condition Name

This parameter, of type Object Name, shall contain the name of an Event Condition object to be referenced by the &eventCondition field of the newly created Event Enrollment.

21.2.1.1.3 Event Condition Transitions

The Event Condition Transitions parameter, of type Transitions, shall specify the set of transitions of the Event Condition for which invocation of the EventNotification service is requested. The allowed values for this field are specified in 21.1.1.

21.2.1.1.4 Alarm Acknowledgement Rule

The Alarm Acknowledgement Rule parameter, of type AlarmAckRule, shall specify the value of the Event Enrollment object's &aaRule field. The allowed values for this field are given in 21.1.1.

21.2.1.1.5 Event Action Name

This optional parameter, of type Object Name, shall specify the value of the &name field of an Event Action object representing an action to be executed when the specified transition of the Event Condition object occurs. The result of this execution shall be included in EventNotification requests initiated as a result of this Event Enrollment.

21.2.1.1.6 Client Application

The semantics and value of this optional parameter, of type Application Reference, are specified in 21.1.1. If included, it shall specify a client application, which may be the requesting MMS client or a third-party application, to be enrolled for the receipt of Event Notifications resulting from the newly created Event Enrollment.

This parameter shall not be specified unless the **tpy** conformance block is supported.

21.2.1.1.6.1 Display Enhancement

The parameter may be present only if the **cspi** CBB has been negotiated. If this parameter is included, one of the following parameters shall be selected.

21.2.1.1.6.1.1 String

This choice indicates that the string form of the Display Enhancement parameter is present. This selection may be made only if the **des** CBB has been negotiated.

21.2.1.1.6.1.2 Index

This choice indicates that the numeric form of the Display Enhancement parameter is present. This selection may be made only if the **dei** CBB has been negotiated.

21.2.1.1.6.1.3 No Enhancement

This choice specifies that no Display Enhancement is present. This parameter shall be selected if neither **des** nor **dei** has been negotiated.

NOTE This choice may also be made (user option) if either **des** or **dei** has been negotiated.

21.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

21.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

21.2.1.3.1 Object Not Defined

This parameter, of type Object Name, shall be present if the error pertains to the non-existence of the Event Condition object specified by the Event Condition Name parameter or the Event Action object specified by the Event Action Name parameter. If the Event Condition object does not exist, this parameter shall contain the name of the Event Condition Name parameter. If the Event Condition object exists but the Event Action object does not exist, this parameter shall contain the Event Action Name parameter. Otherwise, this parameter shall not be present.

21.2.2 Service Procedure

21.2.2.1 Preconditions

The MMS server shall perform the following actions:

a) verify that no Event Enrollment object exists with the same name as the Event Enrollment Name parameter;

b) verify the existence of an Event Condition object with the same name as the Event Condition Name parameter;

c) verify the existence of an Event Action object with the same name as the Event Action Name parameter, if present;

d) verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

e) If the Client Application parameter is present and different from the MMS client requesting the Define Event Enrollment service, verify that the Event Condition Name parameter does not reference an Event Condition object of AA-scope;

f) if the Client Application parameter is present and different from the MMS client requesting the Define Event Enrollment service, verify that the Event Action Name parameter (if present) does not reference an Event Action object of AA-scope.

If either the Event Condition object or the Event Action object does not exist, the MMS server shall issue a Result(-) response with the Error Class of ACCESS, Error Code of OBJECT-NON-EXISTENT, and Object Not Defined parameter containing the value of the name of the object that did not exist.

If any other condition is not satisfied, the MMS server shall issue a Result(-) response with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED.

21.2.2.2 Actions

The MMS server shall create a notification Event Enrollment object initialized as follows:

a) &name - initialized to the value provided in the Event Enrollment Name parameter;

b) &accessControl - initialized to refer to an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) &eeClass - initialized to the value **notification**.

d) &eventCondition - initialized to reference the Event Condition object identified by the value of the Event Condition Name parameter.

e) &ecTransitions - initialized to the value of the Event Condition Transitions parameter.

f) &aAssociation - The value of the &aAssociation field shall be dependent on the presence or absence of the Client Application parameter. The following rules shall apply to determine the value of the &aAssociation field:

   1) If the Client Application parameter is not present in the service request, this field shall be initialized to a value identifying the application association over which the DefineEventEnrollment request was received.

   2) If the Client Application parameter is present in the service request and the client to be enrolled is the MMS client requesting the DefineEventEnrollment service, this field shall be initialized to a value identifying the application association over which the DefineEventEnrollment request was received.

   3) If the Client Application parameter is present in the service request, and the client to be enrolled is not the MMS client requesting the DefineEventEnrollment service and an application association to this Client Application exists, this field shall be initialized to a value identifying the application association with this Client Application.

   4) If the Client Application parameter is present in the service request, and the client to be enrolled is not the MMS client requesting the DefineEventEnrollment service and an application association to this Client Application does not exist, this field shall be undefined.

g) &notificationLost - initialized to the value false.

h) &eventAction - if the Event Action Name parameter is present in the service request, this field shall be initialized to reference the Event Action object identified by the value of the Event Action Name parameter. Otherwise, it shall be undefined.

i) &duration - if the Client Application parameter was specified, this attribute shall be initialized to the value **permanent**. Otherwise, it shall be initialized to the value **current**.

j) &clientApplication - initialized to the value of the Client Application parameter, if specified. Otherwise this field shall be initialized to identify the client application for the application association over which this request was received.

k) &aaRule - initialized to the value of the Alarm Acknowledgement Rule parameter.

l) &timeActiveAck - initialized to the value **undefined**.

m) &timeIdleAck - initialized to the value **undefined**.

n) &ackState - initialized to **acked**.

o) If the Display Enhancement parameter is present and the String choice of Display Enhancement parameter has been selected, the **text** choice of the &displayEnhancement field of the Event Condition object shall be selected and its value shall be the value of String parameter.

p) If the Display Enhancement parameter is present and the Index choice of the Display Enhancement parameter has been selected, the **number** choice of &displayEnhancement field of the Event Condition object shall be selected and its value shall be the value of the Index parameter.

q) If the Display Enhancement parameter is present and the No Enhancement choice of the Display Enhancement parameter has been selected, the **none** choice of &displayEnhancement field of the Event Condition object shall be selected.

A Result(+) shall be returned, indicating that the Event Condition object has been created.

21.3 DeleteEventEnrollment service

The DeleteEventEnrollment service is used by an MMS client to request that an MMS server delete one or more **notification** Event Enrollment objects.

21.3.1 Structure

The structure of the component service primitives is shown in Table 116.

Table 116 - DeleteEventEnrollment service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Delete	M	M(=)			
List of Event Enrollment Names	S	S(=)			
Event Condition Name	S	S(=)			
Event Action Name	S	S(=)			
Result(+)			S	S(=)	
Candidates Not Deleted			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

21.3.1.1 Argument

This parameter shall convey the parameters of the DeleteEventEnrollment service request.

21.3.1.1.1 Scope Of Delete

The Scope Of Delete parameter shall specify the extent of delete that is requested. The Scope of Delete is indicated by the selection of one of the following parameters.

21.3.1.1.2 List of Event Enrollment Names

If this parameter is selected, it shall specify a list of one or more Event Enrollment objects designated as candidates for deletion.

21.3.1.1.3 Event Condition Name

If this parameter is selected, this parameter, of type Object Name, shall specify the value of the &name field of the Event Condition object whose &EventEnrollments field is used to identify the Event Enrollment objects to be considered for deletion. Of these Event Enrollment objects, only those whose &clientApplication field references the requesting MMS client are designated as candidates for deletion.

21.3.1.1.4 Event Action Name

If this parameter is selected, this parameter, of type Object Name, shall specify the value of the &name field of the Event Action object whose &EventEnrollments field is to be used to identify the Event Enrollment objects to be considered for deletion.

21.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall include the following parameter.

21.3.1.2.1 Candidates Not Deleted

This parameter, of type integer, shall contain a count of the number of Event Enrollment objects that were included in the scope of Event Enrollment objects to be deleted, but that were not deleted because the conditions required by the service procedure were not met.

21.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

21.3.2 Service Procedure

21.3.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = DELETE. If this condition is not satisfied, a Result(-) shall be returned with an error class = ACCESS and error code = OBJECT-ACCESS-DENIED.

If the Scope of Delete parameter is Event Condition Name, the MMS server shall verify that the referenced Event Condition object exists. If this object does not exist, a Result(-) response shall be returned and the service procedure terminated.

If the Scope of Delete parameter is Event Action Name, the MMS server shall verify that the referenced Event Action object exists. If this object does not exist, a Result(-) response shall be returned and the service procedure terminated.

The Candidates Not Deleted parameter shall be initialized to zero.

21.3.2.2 Action Step 1

The MMS server shall prepare a list of candidate Event Enrollments for deletion. The objects on this list depend on the selection of the Scope Of Delete parameter.

a) If the Scope of Delete parameter is the List of Event Enrollment Names, that list provides the list of candidate Event Enrollments. For each element of that list, the MMS server shall verify

   1) that the Event Enrollment object exists and

   2) that the conditions in the Access Control List referenced by the Reference to Access Control List attribute of the Event Enrollment object are satisfied for Service Class = DELETE.

   If any of these conditions is not met, this Event Enrollment object shall not be included on the list of Event Enrollment objects and the Candidates Not Deleted parameter shall be increased by one.

b) If the Scope of Delete parameter is Event Condition Name, the &EventEnrollments field of that Event Condition provides the list of candidate Event Enrollments. For each object referenced by the &EventEnrollments field of this Event Condition, the MMS server shall verify:

   1) that the Event Enrollment object exists;

   2) that the conditions in the Access Control List referenced by the &accessControl field of the Event Enrollment object are satisfied for Service Class = DELETE; and

   3) that the &clientApplication field of this Event Enrollment object references the requesting MMS client.

   If any of these conditions is not met, this Event Enrollment object shall not be included on the list of Event Enrollment objects and the Candidates Not Deleted parameter shall be increased by one.

c) If the Scope of Delete parameter is Event Action Name, the &EventEnrollments field of that Event Action object provides the list of candidate Event Enrollments. For each object referenced by the &EventEnrollments field of the Event Action object the MMS server shall verify:

   1) that the Event Enrollment object exists;

   2) that the conditions in the Access Control List referenced by the &accessControl field of the Event Enrollment object are satisfied for Service Class = DELETE; and

   3) that the &clientApplication field of this Event Enrollment object references the requesting MMS client.

   If any of these conditions is not met, this Event Enrollment object shall not be included on the list of Event Enrollment objects and the Candidates Not Deleted parameter shall be increased by one.

21.3.2.3 Action Step 2

For each object of the list of Event Enrollment objects, the MMS server shall execute the Procedure for Event Enrollment Deletion (see 21.3.3). A Result(+) response shall be issued, indicating the number of candidate Event Enrollment objects that were not deleted.

21.3.3 Procedure for Event Enrollment Deletion

The MMS server shall remove the reference to this Event Enrollment object from the &EventEnrollments field of the Event Condition object referenced by the &eventCondition field of this Event Enrollment object.

If the &event Action field of this Event Enrollment object references an Event Action object, the MMS server shall remove the reference to this Event Enrollment object from the &EventEnrollments field of the Event Action object referenced by the &eventAction field of this Event Enrollment object.

The MMS server shall remove the reference to this Event Enrollment object from the &EventEnrollments field of the Access Control List object referenced by the &accessControl field of this Event Enrollment object.

NOTE 1 If an Event Enrollment object having the value **permanent** for the &duration field is deleted, and if the MMS server was the Calling MMS-user on the application association being used to convey event notifications for this Event Enrollment, the MMS server may, as a local matter, issue the Conclude service on that application association.

NOTE 2 This procedure should also be executed for deletion of Event Enrollment objects that result from deletion of a Domain or loss of an application association.

21.4 GetEventEnrollmentAttributes service

The GetEventEnrollmentAttributes service is used by an MMS client to request that an MMS server return the values of descriptive attributes of an Event Enrollment object or a list of Event Enrollment objects that satisfy a specified set of criteria.

21.4.1 Structure

The structure of the component service primitives is shown in Table 117.

Table 117 - GetEventEnrollmentAttributes service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Scope of Request	M	M(=)			
List of Event Enrollment Names	S	S(=)			
Client Application	S	S(=)			
Event Condition Name	S	S(=)			
Client Application	U	U(=)			
Event Action Name	S	S(=)			
Client Application	U	U(=)			
Continue After	U	U(=)			
Result(+)			S	S(=)	
List of EE Attributes			M	M(=)	
Event Enrollment Name			M	M(=)	
Event Condition Name			M	M(=)	
Event Action Name			C	C(=)	
Client Application			U	U(=)	
MMS Deletable			M	M(=)	
Enrollment Class			M	M(=)	
Duration			C	C(=)	
Invode ID			C	C(=)	
Remaining Acceptable Delay			C	C(=)	
Display Enhancement			C	C(=)	cspi
String			S	S(=)	des
Index			S	S(=)	dei
No Enhancement			S	S(=)	
Access Control List			M	M(=)	aco
More Follows			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

21.4.1.1 Argument

This parameter shall convey the parameters of the GetEventEnrollmentAttributes service request.

21.4.1.1.1 Scope Of Request

This parameter shall indicate the scope of Event Enrollment objects to be included in the request. The Scope Of Request is indicated by the selection of one of the following parameters.

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, this parameter value shall be same as specified in the original request.

NOTE The service parameters have been altered from those of the first edition of MMS in order to correspond more closely with the object model. The change is reflected in a new set of derivation rules in part 2 to relate these service parameters to the same protocol.

21.4.1.1.2 List of Event Enrollment Names

If selected, this parameter shall contain a list of values for the &name field of the Event Enrollment objects for which the attributes are desired.

21.4.1.1.3 Client Application

If selected, this parameter, of type Application Reference, shall indicate the specific client application for which the request is made.

21.4.1.1.4 Event Condition Name

If selected, this parameter, of type Object Name, shall contain the value of the &name field of the Event Condition object whose &EventEnrollments field provides the list of Event Enrollment objects that shall be reported.

21.4.1.1.4.1 Client Application

If included, this parameter, of type Application Reference, limits the results to those Event Enrollment objects specified in the &EventEnrollments field of the Event Condition object for which the specified client is enrolled.

21.4.1.1.5 Event Action Name

If selected, this parameter, of type Object Name, shall contain the value of the &name field of the Event Action object whose &EventEnrollments field provides the list of Event Enrollment objects that shall be reported.

21.4.1.1.5.1 Client Application

If included, this parameter, of type Application Reference, limits the results to those Event Enrollment objects specified in the &EventEnrollments field of the Event Action object for which the specified client is enrolled.

21.4.1.1.6 Continue After

If this is a request to continue a previously issued request for which the Result(+) parameter of the confirm primitive specified the value of the More Follows parameter equal to true, this parameter shall be specified. Otherwise, it shall be omitted. If specified, this parameter shall contain the value of the Event Enrollment Name parameter, of type Object Name, from the last entry in the List Of EE Attributes parameter taken from the confirm primitive for which continuation is desired.

21.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

21.4.1.2.1 List of EE Attributes

The List of EE Attributes parameter shall contain a list composed of descriptive attributes of zero or more Event Enrollment objects satisfying the criteria of the request primitive. Each entry in the list shall contain the following parameters.

21.4.1.2.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the Event Enrollment object's &name field.

21.4.1.2.1.2 Event Condition Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object. If the referenced Event Condition object has become unavailable, for example, as a result of deletion of a Domain or loss of an application association, this parameter shall have the value UNDEFINED.

21.4.1.2.1.3 Event Action Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Action object referenced by the Event Enrollment object's &eventAction field if present. If the &eventAction field is not present in the Event Enrollment object, this parameter shall be omitted. If the &eventAction field is present but the Event Action object has become unavailable, for example, as a result of deletion of a Domain or loss of an application association, this parameter shall have the value UNDEFINED, and shall be included. If the &eeClass field of the Event Enrollment object is **modifier**, this parameter shall be omitted.

21.4.1.2.1.4 Client Application

If the &clientApplication field of the Event Enrollment does not specify the requesting MMS client, this parameter, of type Application Reference, shall contain the value of the Event Enrollment object's &clientApplication field. Otherwise, this parameter shall be omitted. If the &eeClass field of the Event Enrollment object is **modifier**, this parameter shall be omitted.

21.4.1.2.1.5 MMS Deletable

This parameter, of type boolean, shall indicate whether (true) or not (false) the Event Enrollment object may be deleted using the DeleteEventEnrollment service. Subclause 9.1.4 specifies the value to be returned by this parameter.

21.4.1.2.1.6 Enrollment Class

This parameter, of type EE-Class, shall contain the value of the Event Enrollment object's &eeClass field.

21.4.1.2.1.7 Duration

This parameter, of type EE-Duration, shall contain the value of the Event Enrollment object's &duration field. If the &eeClass field of the Event Enrollment object does not contain the value **notification**, this parameter shall be omitted.

21.4.1.2.1.8 Invoke ID

This parameter, of type integer, shall contain the value of the &invokeID field of the Event Enrollment object. If the &eeClass field of the Event Enrollment object does not contain the value **modifier**, this parameter shall be omitted.

21.4.1.2.1.9 Remaining Acceptable Delay

This optional parameter, of type integer, shall contain the value of the &remainingDelay field of the Event Enrollment object if present. If the &eeClass field of the Event Enrollment object contains the value **modifier** and this parameter is not present, an unbounded delay is implied. If the &eeClass field of the Event Enrollment object does not contain the value **modifier**, this parameter shall be omitted.

21.4.1.2.1.10 Display Enhancement

This parameter shall contain the value of the &displayEnhancement field of the Event Condition object. This parameter shall not appear unless the **cspi** CBB has been negotiated. Depending on its value, one of the following parameters shall be selected.

21.4.1.2.1.10.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter. This parameter may not be selected unless the **des** CBB has been negotiated.

21.4.1.2.1.10.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This parameter may not be selected unless the **dei** CBB has been negotiated.

21.4.1.2.1.10.3 No Enhancement

This parameter, of type null, specifies that no Display Enhancement is present.

21.4.1.2.1.11 Access Control List

This parameter, of type Identifier, shall indicate the &name field of the Access Control List object that controls access to this Event Enrollment object. This parameter shall not appear unless the **aco** parameter CBB has been negotiated.

21.4.1.2.2 More Follows

This parameter, of type boolean, shall have the value true if this response does not contain attribute values from the last of the requested Event Enrollment objects. Otherwise, it shall be false.

21.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

21.4.2 Service Procedure

21.4.2.1 Preconditions

None.

21.4.2.2 Action Step 1

The MMS server shall prepare a list of Event Enrollments whose attributes are to be returned. The extent of this list depends on the value of the Scope Of Request parameter. In each of the service procedures (specified below) the number of Event Enrollment objects whose attributes may be returned in the List Of EE Attributes parameter in a single service response may be limited by local restrictions. If the response does not contain the last of the requested Event Enrollment objects, the More Follows parameter shall be set to the value true in the response primitive. Otherwise, the value of the More Follows parameter shall be false.

21.4.2.2.1 Scope of Request is List of Event Enrollment Names

The List of Event Enrollment Names parameter specifies the list of the Event Enrollment objects whose attributes are to be returned.

21.4.2.2.2 Scope Of Request is Client Application

The list of Event Enrollment objects whose attributes are to be returned shall be constructed as follows:

a) If the Client Application parameter identifies the MMS client of this service request, for each application association object referenced by the &Associations field of the VMD,

for each of the Event Condition objects referenced by the &EventConditions field of that application association,

for each Event Enrollment object referenced by the &EventEnrollments field of that Event Condition object, if the Client Application parameter matches the &clientApplication field of that Event Enrollment object, include that Event Enrollment object in the list.

b) For each Event Condition object referenced by the &EventConditions field of the VMD,

for each Event Enrollment object referenced by the &EventEnrollments field of that Event Condition object, if the Client Application parameter matches the &clientApplication field of that Event Enrollment object, include that Event Enrollment object in the list.

c) for each Domain object referenced by the &Domains field of the VMD,

for each of the Event Condition objects referenced by the &EventConditions field of that Domain,

for each Event Enrollment object referenced by the &EventEnrollments field of that Event Condition object, if the Client Application parameter matches the &clientApplication field of that Event Enrollment object, include that Event Enrollment object in the list.

The ordering of the Event Conditions of this procedure is prescribed in 5.4.2; the ordering of the Event Enrollments within each Event Condition is also prescribed in 5.4.2.

21.4.2.2.3 Scope Of Request is Event Condition

If the Client Application parameter is specified, the list shall contain the Event Enrollments referenced by the &EventEnrollments field of the Event Condition object specified by the Event Condition Name parameter whose &clientApplication field (of the Event Enrollment) matches the Client Application parameter. If the Client Application parameter is not specified, the list is the entire &EventEnrollments field of the Event Condition object specified by the Event Condition Name parameter.

21.4.2.2.4 Scope Of Request is Event Action

If the Client Application parameter is specified, the list shall contain the Event Enrollments referenced by the &EventEnrollments field of the Event Action object specified by the Event Action Name parameter whose &clientApplication field (of the Event Enrollment) matches the Client Application parameter. If the Client Application parameter is not specified, the list is the entire &EventEnrollments field of the Event Action object specified by the Event Action Name parameter.

21.4.2.3 Action Step 2

If the Continue After parameter is present in the service indication, the MMS server shall begin the response at the Event Enrollment object following the object indicated by the Continue After parameter. Otherwise, the MMS server shall begin at the beginning of the list.

The MMS server shall return the attributes of one or more Event Enrollments from this list. The decision about the number of attribute sets to be included in the response is a local matter. If the MMS server terminates the response before the list is exhausted, it shall include a More Follows parameter of true; otherwise it shall return a More Follows parameter of false. A Result(+) shall be returned.

NOTE In a well formed VMD, there should be no duplicates in the list prepared in this manner. Each Event Enrollment is associated with one and only one Event Condition and/or Event Action. It is a local matter of what action to take if this list contains duplicates. Reinitialization of the system may be required.

21.5 ReportEventEnrollmentStatus service

The ReportEventEnrollmentStatus service is used by an MMS client to obtain the status of a single notification Event Enrollment object.

21.5.1 Structure

The structure of the component service primitives is shown in Table 118.

Table 118 - ReportEventEnrollmentStatus service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Enrollment Name	M	M(=)			
Result(+)			S	S(=)	
Event Condition Transitions			M	M(=)	
Notification Lost			M	M(=)	
Duration			M	M(=)	
Alarm Acknowledgement Rule			C	C(=)	
Current State			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

21.5.1.1 Argument

This parameter shall convey the parameter of the ReportEventEnrollmentStatus service request.

21.5.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the &name field of the Event Enrollment object for which the ReportEventEnrollmentStatus service is requested.

21.5.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

21.5.1.2.1 Event Condition Transitions

This parameter, of type Transitions, shall contain the value of the Event Enrollment object's &ecTransitions field.

21.5.1.2.2 Notification Lost

This parameter, of type boolean, shall contain the value of the Event Enrollment object's &notificationLost field.

21.5.1.2.3 Duration

This parameter, of type EE-Duration, shall contain the value of the Event Enrollment object's &duration field.

21.5.1.2.4 Alarm Acknowledgement Rule

This parameter, of type AlarmAckRule, shall contain the value of the Event Enrollment object's &aaRule field.

21.5.1.2.5 Current State

This parameter, of type EE-State, depends on the value of the &aaRule field of the Event Enrollment object and shall reflect the value of the &ecState field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object and the value of the &ackState field of the Event Enrollment object. The value of this parameter is determined as follows:

a) if the value of the &ecState field is **disabled**, the parameter shall be DISABLED;

b) if the value of the &ecState field is **active**,

1) if the value of the &aaRule field is **none**, the parameter shall be ACTIVE;

2) otherwise,

i) if the value of the &ackState field is **noAckA**, the parameter shall be ACTIVE-NO-ACK-A;

ii) if the value of the &ackState field is **acked**, the parameter shall be ACTIVE-ACKED;

c) if the value of the &ecState field is **idle**,

1) if the value of the &aaRule field is **none** or **simple**, the parameter is IDLE;

2) if the value of the &aaRule field is **ack-active**,

i) if the value of the &ackState field is **noAckA**, the parameter is IDLE-NO-ACK-A;

ii) if the value of the &ackState field is **acked**, the parameter is IDLE-ACKED;

3) if the value of the &aaRule field is **ack-all**,

i) if the value of the &ackState field is **noAckI**, the parameter is IDLE-NO-ACK-I;

ii) if the value of the &ackState field is **noAckA**, the parameter is IDLE-NO-ACK-A;

i) if the value of the &ackState field is **acked**, the parameter is IDLE-ACKED.

If the Event Condition object has become undefined, as a result of a deletion of a Domain or loss of application association, the value of the &lastState field of the Event Enrollment object shall be used in place of the &ecState field in the determination of this parameter.

21.5.1.3 Result(-)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

21.5.2 Service Procedure

21.5.2.1 Preconditions

If the Event Enrollment object specified by the Event Enrollment Name parameter does not exist, the MMS server shall return a Result(-) response.

21.5.2.2 Actions

The MMS server shall issue a Result(+) response, containing the current values for the specified Event Enrollment object's attributes and derived parameters.

21.6 AlterEventEnrollment service

The AlterEventEnrollment service is used by an MMS client to request that the MMS server replace the value of the &ecTransitions field, the value of the &aaRule field, the value of the &displayEnhancement field, or two or more of these fields of an existing **notification** Event Enrollment object.

21.6.1 Structure

The structure of the component service primitives is shown in Table 119.

Table 119 - AlterEventEnrollment service

Parameter Name	Req	Ind	Rep	Cnf	CBB
Argument	M	M(=)			
Event Enrollment Name	M	M(=)			
Event Condition Transitions	U	U(=)			
Alarm Acknowledgement Rule	U	U(=)			
Display Enhancement	C	C(=)			cspi
String	S	S(=)			des
Index	S	S(=)			dei
No Enhancement	S	S(=)			
Result(+)			S	S(=)	
Current State			M	M(=)	
Transition Time			M	M(=)	
Result(-)			S	S(=)	
Error Type			M	M(=)	

21.6.1.1 Argument

This parameter shall convey the parameters of the AlterEventEnrollment service request.

21.6.1.1.1 Event Enrollment Name

This parameter, of type Object Name, shall contain the value of the &name field of the **notification** Event Enrollment object that is to be modified.

21.6.1.1.2 Event Condition Transitions

The Event Condition Transitions parameter, of type Transitions, shall specify the set of transitions of the Event Condition object for which invocation of the Procedure for Event Transition Processing is requested. The allowed values for this parameter are specified in 21.1.1.

Either this parameter, the Alarm Acknowledgement Rule parameter, the Display Enhancement parameter, or two or more, shall be specified.

21.6.1.1.3 Alarm Acknowledgement Rule

The Alarm Acknowledgement Rule parameter, of type AlarmAckRule, shall specify the new value of the Event Enrollment object's &aaRule field. The semantics and allowed values for this parameter are given in 21.1.1.

This parameter shall not be present if the referenced Event Condition object is a **network-triggered** Event Condition.

Either this parameter, the Event Condition Transitions parameter, the Display Enhancement parameter, or two or more, shall be specified.

21.6.1.1.3.1 Display Enhancement

This parameter indicates that the &displayEnhancement field of the Event Enrollment shall be altered by this service. This parameter shall not appear unless the **cspi** CBB has been negotiated. If this parameter is present, one of the following parameters shall be selected.

Either this parameter, the Event Condition Transitions parameter, the Alarm Acknowledgement Rule parameter, or two or more, shall be specified.

21.6.1.1.3.1.1 String

This parameter, of type character string, is the string form of the Display Enhancement parameter. This selection may be made only if the **des** CBB has been negotiated.

21.6.1.1.3.1.2 Index

This parameter, of type integer, is the numeric form of the Display Enhancement parameter. This selection may be made only if the **dei** CBB has been negotiated.

21.6.1.1.3.1.3 No Enhancement

This parameter, of type NULL, specifies that no Display Enhancement is present. This parameter shall be selected if neither **des** nor **dei** has been negotiated.

NOTE This parameter may also be selected if either **des** or **dei** has been negotiated.

21.6.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result shall return the following parameters.

21.6.1.2.1 Current State

This parameter, of type EE-State, depends on the value of the &aaRule field of the Event Enrollment object and shall reflect the value of the &ecState field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object and the value of the &ackState field of the Event Enrollment object. The value of this parameter is specified in 21.5.1.2.5.

21.6.1.2.2 Transition Time

This parameter shall contain the time (date and time of day or Time Sequence Identifier) at which the transition to the current value of the &ackState of the Event Enrollment occurred. If execution of this procedure resulted in an alteration of the value of the &ackState field, this parameter shall contain the time at which the alteration occurred. Otherwise, this parameter shall be equal to the value of the &timeToIdle field or the value of the &timeToActive field of the Event Condition object referenced by the &eventCondition field of the Event Enrollment object, whichever is later. If the value of the &eventCondition field of the Event Enrollment object is undefined, the value of this parameter shall be UNDEFINED.

21.6.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

21.6.2 Service Procedure

21.6.2.1 Preconditions

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the VMD are satisfied for the service class = LOAD.

The MMS server shall verify that all the conditions in the Access Control List object referenced by the &accessControl field of the Event Enrollment object are satisfied for the service class = LOAD.

If any of these conditions is not satisfied, the service shall fail and a Result(-) shall be returned.

21.6.2.2 Actions

The MMS server shall replace the value of its &ecTransitions field, the value of its &aaRule field, or the value of the &displayEnhancement field, or two or all of these fields, with the values specified in the Event Condition Transitions, Alarm Acknowledgement Rule, and Display Enhancement parameters, respectively.

If the value of the Alarm Acknowledgement Rule parameter is not equal to the value of the &aaRule field prior to this service procedure, the &ackState field of the Event Enrollment object may be altered. Following the change to the &aaRule field, the &ackState field of the Event Enrollment object shall be determined as follows:

a) If the &aaRule field is **none** or **simple**, set the &ackState field of the Event Enrollment to **acked**.

b) If the &aaRule field is **ack-active**, and the &ecState field of the referenced Event Condition object is **active** or **idle**,

   1) If the &timeActiveAck field is **undefined**, set the &ackState field of the Event Enrollment object to **noAckA**;

   2) Otherwise, set the &ackState field of the Event Enrollment object to **acked**.

c) If the &aaRule field is **ack-all**,

   1) If the &timeActiveAck field is **undefined** and the &ecState field of the referenced Event Condition object is **active** or **idle**, set the &ackState field of the Event Enrollment object to **noAckA**;

   2) Otherwise, if the &timeIdleAck field is **undefined** and the &ecState field of the referenced Event Condition object is **idle**, set the &ackState field of the Event Enrollment object to **noAckI**;

   3) Otherwise, set the &ackState field of the Event Enrollment object to **acked**.

If the Display Enhancement parameter is present, the value of the &displayEnhancement field of the Event Enrollment object shall be altered. If String is selected, the &displayEnhancement field of the Event Enrollment shall be set to the value of the String parameter. If Index is selected, the &displayEnhancement shall be set to the value of the Index parameter. If No Enhancement is selected, the &displayEnhancement shall be set to **none**.

The MMS server shall return a Result(+) response containing the Current State and Transition Time parameters.

22 Event Condition List services

22.1 Event Condition Lists

This clause provides an object model for the following object:

EVENT-CONDITION-LIST

This clause specifies the following services:

DefineEventConditionList
DeleteEventConditionList
AddEventConditionListReference
RemoveEventConditionListReference
GetEventConditionListAttributes
ReportEventConditionListStatus
AlterEventConditionListMonitoring

The Event Condition List object shall be used to reference groups of Event Condition objects that are required to be operated on as groups. Support of the **recl** CBB (see 8.1.3.20.3) indicates support for Event Condition Lists that may refer to other Event Condition Lists. If this CBB is not supported, the Event Condition List may contain only references to Event Conditions.

22.1.1 The Event Condition List object

```
EVENT-CONDITION-LIST ::= CLASS {
    &name                                ObjectName,
-- shall be unique within its range of specification (VMD, Domain, AA)
    &accessControl                       ACCESS-CONTROL-LIST,
    &EventConditions                     EVENT-CONDITION,
IF (recl)
    &EventConditionLists                 EVENT-CONDITION-LIST,
    &ReferencingEventConditionLists      EVENT-CONDITION-LIST
ENDIF
    }
```

22.1.1.1 &name

The &name field uniquely identifies the Event Condition List object within the VMD. An Event Condition List name may have VMD, Domain-specific or AA-specific scope.

22.1.1.2 &accessControl

This field, of type boolean, shall indicate whether (true) or not (false) this object may be deleted through the use of the DeleteEventConditionList service.

22.1.1.3 &EventConditions

This field identifies a set of zero or more Event Condition objects.

Because of visibility constraints, if the &name field of the Event Condition List object has VMD-specific or Domain-specific scope, this field shall only include Event Condition objects that have VMD-specific or Domain-specific scope. If the &name field of the Event Condition List object has AA-specific scope, this field may include Event Condition objects of any scope.

22.1.1.4 &EventConditionLists

This field identifies a set of Event Condition List objects that are hierarchically subordinate to this Event Condition List object. This field may be present only if the **recl** parameter CBB has been negotiated. This field shall not contain circular references.

Because of visibility constraints, if the &name field of the Event Condition List object has VMD-specific or Domain-specific scope, this field shall only include Event Condition List objects that have VMD-specific or Domain-specific scope. If the &name field of the Event Condition List object has AA-specific scope, this field may include Event Condition List objects of any scope.

22.1.1.5 &ReferencingEventConditionLists

This field identifies other Event Condition List objects that reference this specific Event Condition List object. This field may be present only if the **recl** parameter CBB has been negotiated.

NOTE This field is necessary to fully describe the service procedures that operate on the Event Condition List object. This field is not visible nor modifiable by MMS services.

22.2 DefineEventConditionList service

The DefineEventConditionList service is used by an MMS server to request the MMS server to create an Event Condition List object.

22.2.1 Structure

The structure of the component service primitives is shown in Table 120.

Table 120 - DefineEventConditionList service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List Name	M	M(=)			
List of Event Condition Names	M	M(=)			
List of Event Condition List Names	C	C(=)			recl
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	
Object in error			C	C(=)	

22.2.1.1 Argument

This parameter shall convey the parameters of the DefineEventConditionList service request.

22.2.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the name of the Event Condition List object that is to be created.

22.2.1.1.2 List of Event Condition names

This parameter shall identify a list of Event Condition objects to be included in the specified Event Condition List. This list shall not be empty if **recl** has not been negotiated. If the scope of the Event Condition List Name parameter is VMD-specific or Domain-specific, this parameter shall not contain an Event Condition Name whose scope is AA-specific.

22.2.1.1.3 List of Event Condition List names

This parameter shall identify a list of Event Condition List objects to be included, by reference to each included object, in the Event Condition List. This parameter shall not be present if the **recl** CBB has not been negotiated. If the scope of the Event Condition List Name parameter is VMD-specific or Domain-specific, this parameter shall not contain an Event Condition List Name whose scope is AA-specific.

22.2.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

22.2.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

22.2.1.3.1 Object in error

This parameter, of type Object Name, shall be present if the error concerns the nonexistence or inconsistency of an Event Condition object specified in the List of Event Condition Names parameter, or an Event Condition List object specified in the List of Event Condition List Names parameter. It shall provide the name of the object that caused the error at the VMD. This parameter shall not be present if the failure of this service is not due to the nonexistence or inconsistency of an Event Condition object or a Event Condition List object.

22.2.2 Service procedure

22.2.2.1 Preconditions

The MMS server shall verify that no other Event Condition List objects exist at the VMD with a &name field equal to the value of the Event Condition List Name parameter. Otherwise the MMS server shall issue a Result(-) response.

If any of the Event Condition objects specified in the List of Event Condition Names parameter does not exist at the VMD, or if any of the Event Condition List objects specified in the List of Event Condition List Names parameter does not exist at the VMD, the MMS server shall issue the Result(-) response with an Error Class of ACCESS, Error Code of OBJECT-NON-EXISTENT, and the Object in error parameter.

If the scope of the Event Condition List Name parameter is VMD-specific or Domain-specific, and the scope of the name of any of the Event Conditions in the List of Event Condition Names parameter or of any of the Event Condition Lists in the List of Event Condition List Names parameter is AA-specific, the MMS server shall issue a Result(-) response with an Error Class of DEFINITION, Error Code of OBJECT-ATTRIBUTE-INCONSISTENT, and the Object in error parameter.

22.2.2.2 Actions

The MMS server shall create a new Event Condition List object and initialize it as follows:

a) &name - initialized to the value of the Event Condition List name parameter;

b) &accessControl - initialized to reference an Access Control List object that will report the value of MMS Deletable as true (see 9.1.4). The predefined symbol 'M_Deletable' (see 25.3.2.1) may be used for this purpose.

c) &EventConditions - initialized to refer to the Event Condition objects specified by the value of the List of Event Condition Names parameter.

d) &EventConditionLists - initialized to refer to the Event Condition List objects specified by the value of the List of Event Condition List Names parameter, if present.

e) &ReferencingEventConditionLists - initialized to an empty list.

If the List of Event Condition Names parameter is not empty, for every Event Condition Object specified in the List of Event Condition Names parameter, the MMS server shall place a reference to the newly created Event Condition List object in the Event Condition object's &ReferencingEventConditionLists field.

If the List of Event Condition List Names parameter is not empty, for every Event Condition List object specified in the List of Event Condition List Names parameter, the MMS server shall place a reference to the newly created Event Condition List object in the referenced Event Condition List object's &ReferencingEventConditionLists field.

A Result(+) response shall be issued, indicating that the Event Condition List object was created.

22.3 DeleteEventConditionList service

The DeleteEventConditionList service is used by an MMS client to request the MMS server to delete an Event Condition List object.

22.3.1 Structure

The structure of the component service primitives is shown in Table 121.

Table 121 - DeleteEventConditionList service

Parameter name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List Name	M	M(=)			
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	

22.3.1.1 Argument

This parameter shall convey the parameters of the DeleteEventConditionList service request.

22.3.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the &name field of the Event Condition List object that is to be deleted.

22.3.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

22.3.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, provides the reason for failure.

22.3.2 Service procedure

22.3.2.1 Preconditions

The MMS server shall verify:

a) that all the conditions in the Access Control List referenced by the &accessControl field of the VMD are satisfied for the service class DELETE (see 9.1.3);

b) that the Event Condition List object identified by the Event Condition List Name parameter exists;

c) that all the conditions in the Access Control List referenced by the &accessControl field of the Event Condition List object are satisfied for the service class DELETE (see 9.1.3);

d) that the &ReferencingEventConditionLists field (if present) of the specified Event Condition List object is empty.

If any of these conditions is not satisfied, a Result(-) shall be returned.

22.3.2.2 Actions

For each Event Condition object referenced in the &EventConditions field of the Event Condition List object, the MMS server shall remove the reference to the specified Event Condition List object from the &ReferencingEventConditionLists field of the Event Condition object.

For each Event Condition List object referenced in the &EventConditionLists field (if present) of the Event Condition List object, the MMS server shall remove the reference to the specified Event Condition List object from the &ReferencingEventConditionLists field of the Event Condition List object.

The MMS server shall remove the reference to the specified Event Condition List object from the &EventConditionLists field of the Access Control List object referenced by the &accessControl field of the Event Condition List object.

The MMS server shall delete the specified Event Condition List object, and return a Result(+) response.

22.4 AddEventConditionListReference service

The AddEventConditionListReference service is used by the MMS client to request the MMS server to add an Event Condition object references, or Event Condition List object references, or both to an Event Condition List object.

22.4.1 Structure

The structure of the component service primitives is shown in Table 122.

Table 122 - AddEventConditionListReference service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List Name	M	M(=)			
List of Event Condition Names	M	M(=)			
List of Event Condition List Names	C	C(=)			recl
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	
Object in error			C	C(=)	

22.4.1.1 Argument

This parameter shall convey the parameters of the AddEventConditionListReference service request.

22.4.1.1.1 Event Condition List Name

This parameter, of type Object Name, shall specify the &name field of the Event Condition List object that is to be modified.

22.4.1.1.2 List of Event Condition Names

This parameter shall identify a list of Event Condition objects to be added to the specified Event Condition List. This list shall not be empty if **recl** has not been negotiated. If the scope of the Event Condition List Name parameter is VMD-specific or Domain-specific, this parameter shall not contain an Event Condition Name whose scope is AA-specific.

22.4.1.1.3 List of Event Condition List Names

This parameter shall identify a list of Event Condition List objects to be added to the Event Condition List. This parameter shall not be present if the **recl** CBB has not been negotiated. If the scope of the Event Condition List Name parameter is VMD-specific or Domain-specific, this parameter shall not contain an Event Condition List Name whose scope is AA-specific.

22.4.1.2 Result(+)

The Result(+) parameter shall indicate that the service request succeeded. A successful result does not return service specific parameters.

22.4.1.3 Result(-)

The Result(-) parameter shall indicate that the service request failed. The Error Type parameter, which is defined in detail in clause 24, shall provide the reason for failure. When failure is indicated, the following parameter shall be returned.

22.4.1.3.1 Object in error

This parameter, of type Object Name, shall be present if the error concerns the nonexistence or inconsistency of an Event Condition object specified in the List of Event Condition Names parameter, or an Event Condition List object specified in the List of Event Condition List Names parameter. It shall provide the name of the object that caused the error. This parameter shall not be present if the failure of this service is not due to the nonexistence or inconsistency of an Event Condition object or a Event Condition List object.

22.4.2 Service procedure

22.4.2.1 Preconditions

The MMS server shall verify:

a) that all the conditions in the Access Control List referenced by the &accessControl field of the VMD are satisfied for the service class LOAD (see 9.1.3);

b) that the Event Condition List object identified by the Event Condition List Name parameter exists;

c) that all the conditions in the Access Control List referenced by the &accessControl field of the Event Condition List object are satisfied for the service class LOAD (see 9.1.3);

d) that all the Event Condition objects specified by the List of Event Condition Names parameter exist;

e) that all the Event Condition List objects specified by the List of Event Condition List Names parameter (if present) exist;

f) if the scope of the &name field of any of the Event Condition in the List of Event Condition Names parameter is AA-specific, that the scope of the &name field of the Event Condition List is also AA-specific;

g) if the scope of the &name field of any of the Event Condition List in the List of Event Condition List Names parameter (if present) is AA-specific, that the scope of the &name field of the Event Condition List is also AA-specific;

If the Event Condition List object specified by the Event Condition List Name parameter does not exist, a Result(-) response shall be issued with Error Class ACCESS and Error Code OBJECT-NON-EXISTENT without an Object in error parameter.

If any of the Event Condition objects specified in the List of Event Condition Names parameter does not exist, or if any of the Event Condition List objects specified in the List of Event Condition List Names parameter does not exist, a Result(-) response shall be issued with Error Class ACCESS, Error Code OBJECT-NON-EXISTENT, and the Object in error parameter.

If the scope of the Event Condition List Name parameter is VMD-specific or Domain-specific, and the scope of the &name field of any of the Event Conditions in the List of Event Condition Names parameter or of any of the Event Condition Lists in the List of Event Condition List Names parameter is AA-specific, a Result(-) response shall be issued with an Error Class of DEFINITION, Error Code of OBJECT-ATTRIBUTE-INCONSISTENT, and the Object in error parameter.

Otherwise, if any other of these conditions is not satisfied, a Result(-) shall be returned without an Object in error parameter.

22.4.2.2 Action Step 1

For every Event Condition Object specified in the List of Event Condition Names parameter, the MMS server shall:

a) verify that the Event Condition object is not already included in the &EventConditionLists field of this Event Condition List object; if it is included, skip the remainder of this step for this Event Condition;

b) add a reference to the specified Event Condition List object to the Event Condition object's &ReferencingEventConditionLists field;

c) add a reference to the Event Condition object to the specified Event Condition List object's &EventConditions field.

22.4.2.3 Action Step 2

NOTE In this subclause, the Event Condition List identified by the Event Condition List Name parameter is referred to as the named Event Condition List; each Event Condition List object in the List of Event Condition List Names parameter is referred to as a referenced Event Condition List.

If the List of Event Condition List Names parameter has been provided, for every Event Condition List object specified in the List of Event Condition List Names parameter, the MMS server shall:

a) verify that the referenced Event Condition List object is not already included in the &EventConditionLists field of the named Event Condition List object; if it is included, skip the remainder of this step for this Event Condition List;

b) add a reference to the named Event Condition List object to the referenced Event Condition List object's &ReferencingEventConditionLists field;

c) add a reference to the referenced Event Condition List object to the named Event Condition List object's &EventConditionLists field.

A Result(+) response shall be issued, indicating that the Event Condition List object was modified and references updated.

22.5 RemoveEventConditionListReference service

The RemoveEventConditionListReference service is used by an MMS client to request the MMS server to remove Event Condition object references, or to remove Event Condition List object references, or both, from a specified Event Condition List object.

22.5.1 Structure

The structure of the component service primitives is shown in Table 123.

Table 123 - RemoveEventConditionListReference service

Parameter Name	Req	Ind	Rsp	Cnf	CBB
Argument	M	M(=)			
Event Condition List Name	M	M(=)			
List of Event Condition Names	M	M(=)			
List of Event Condition List Names	C	C(=)			recl
Result(+)			S	S(=)	
Result(-)			S	S(=)	
Error type			M	M(=)	
Object in error			C	C(=)	

22.5.1.1 Argument

This parameter shall convey the parameters of the RemoveEventConditionListReference service request.

22.5.1.1.1 Event Condition List name

This parameter, of type Object Name, shall specify the &name field of the Event Condition List object that is to be modified.

22.5.1.1.2 List of Event Condition Names

This parameter shall identify a list of Event Condition objects to be removed from the specified Event Condition List. This list shall not be empty if **recl** CBB has not been negotiated.